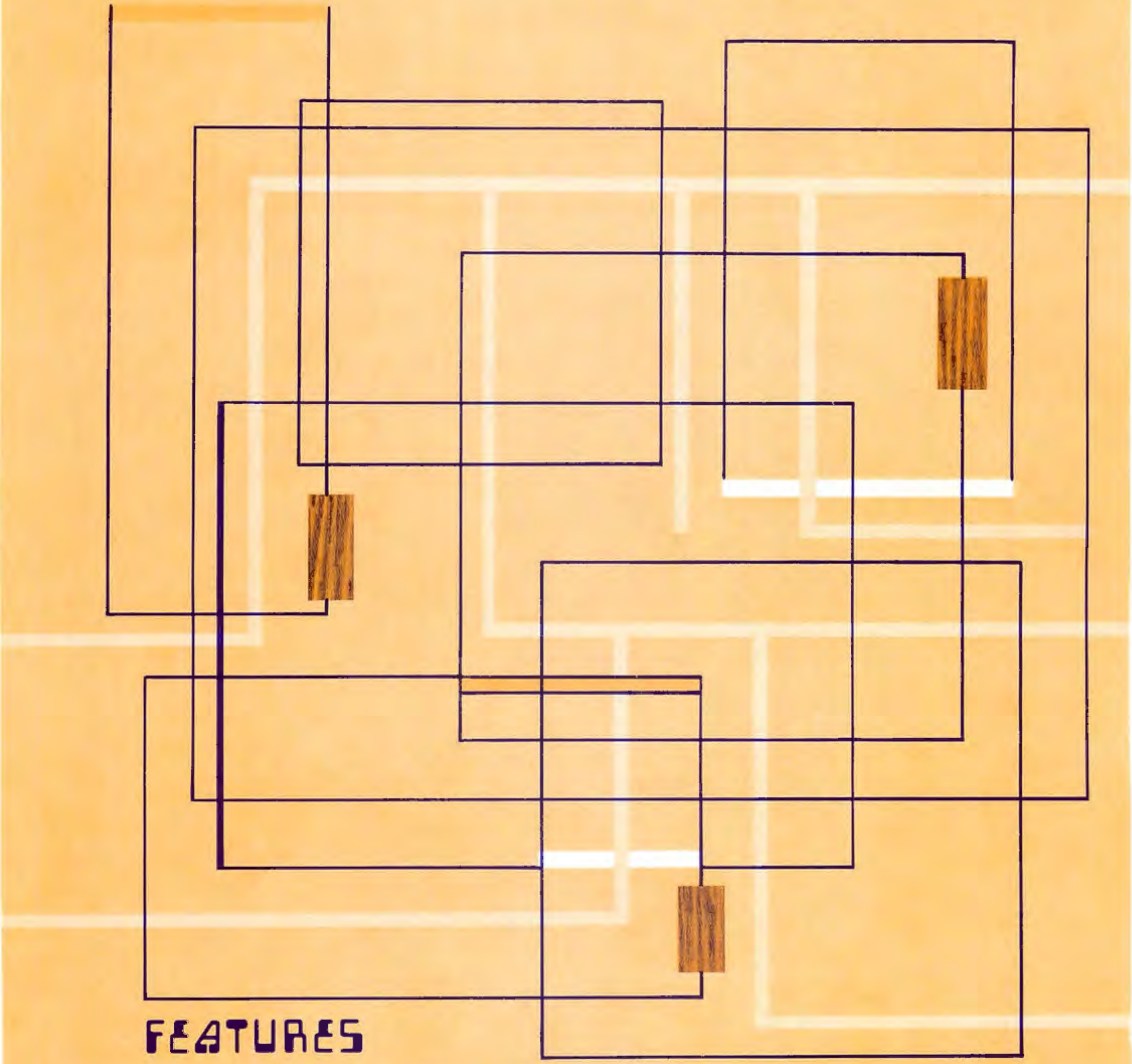
Price $1.25

VOL. 1 ISSUE 2 JANUARY 1976

SCCS INTERFACE

THE OFFICIAL PUBLICATION OF THE SOUTHERN CALIFORNIA COMPUTER SOCIETY



FEATURES

Introduction to Microprocessor Technology

Toward the Understanding of Large Scale Systems

ASCII to OCTAL Machine Language Loader

The First 100 Days

THE SOUTHERN CALIFORNIA COMPUTER SOCIETY

OBJECTIVES

A non-profit organization whose purpose is:

A. To exchange and disseminate information among the Society members concerning the computer arts and sciences.

B. To provide a technical assistance to other members of the Society in those computer projects which are not undertaken for pecuniary gain or profit including but not limited to hardware, software, and computer programming.

C. To publish books, newsletters, magazines, and other periodicals for the benefit and education of the Society members and the general public.

D. To conduct and sponsor seminars, lectures, and courses relating to the computer arts and sciences.

F. To develop and maintain computer centers and laboratory workshops for the members of the Society and the general public including provisions for time-sharing operations.

PUBLICATIONS

SCC INTERFACE, the official publication of the Southern California Computer Society is published monthly. Its content is composed primarily of articles contributed by members of the Society and intended for the reader with an interest in computers for professional or avocational reasons.

MEMBERSHIP

Membership in the Society is open to anyone, regardless of educational background or geographic location. All members are entitled to the rights of meetings, elections, receipt of official publications in a timely manner, and all other benefits as provided by the officers, Board of Directors, and various committees of the Society pursuant to the by-laws of the Southern California Computer Society.

MEMBERSHIP CLASSIFIED ADVERTISING

Each month SCCS Interface will devote some space free, non-commercial advertising by members. This is done as a service to members of the Southern California Computer Society and to help promote communications between computer experimenters and hobbyists throughout the world.

Many SCCS members, being pioneers in the small computer field, often have for sale, on an occasional basis, used computing equipment, or equipment they have designed and built themselves. Those members, as well as those who simply wish to establish contact with individuals with similar or complimentary interests, are encouraged to utilize the member ad privilege subject to the following conditions:

A. Ads must be received by the ad editor of SCCS Interface by the first of the month prior to issue month.

B. Ads may list the member's name, home address and phone. No commercial firm names, please!

C. Ads must be typewritten, double spaced and no more than 250 characters (including spaces and punctuation) in length. Ads 250-500 characters will be printed on space-available basis.

D. Ads may be run up to three consecutive issues, however, a specific request must be received for each instance.

E. No free member ads will be accepted for the sale of equipment or services by a commercial profit-oriented individual or organization.

Address all member ads to:
Interface Ad Editor, Nancy Delong, 8005 Denrock Ave., Los Angeles, Calif. 90045

Those non-members wishing to advertise are invited to join SCCS and enjoy the member ad privilege as well as the delivery of SCCS INTERFACE to your mail box.

COMMERCIAL CLASSIFIED

Commercial ads limited to business card format will be accepted at the rate of $25.00 per ad. Payment must accompany order. Submit orders to:
"Commercial Classified", McPheters, Wolfe & Jones, 6515 Sunset Blvd., Suite 202
Hollywood, California 90028 (213) 469-7789

PUBLISHER

McPHETERS, WOLFE & JONES, 6515 Sunset Blvd., Suite 202, Hollywood, California 90028

All editorial content and effort are contributed at no cost to the Society. Opinions expressed in by-lined articles do not necessarily reflect the opinion of this magazine or the Southern California Computer Society. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes an endorsement of the product or products by this magazine or by the Southern California Computer Society.

DISPLAY ADVERTISING

Direct all advertising inquiries to:

Advertising Department
SCCS INTERFACE - PRODUCTION UNIT
c/o McPHETERS, WOLFE & JONES
P. O. Box 1234
Cerritos, California 90701
(213) 469-7789

CONTRIBUTIONS

They must be accompanied by return postage and we assume no responsibility for loss or damage thereto. Any material accepted is subject to such revision as is necessary in our sole discretion to meet the requirements of this publication. The act of mailing a manuscript and/or material shall constitute an express warranty by the contributor that the material is original and in no way an infringement upon the rights of others.

SCCS INTERFACE
The Official Publication of the Southern California Computer Society
Vol. 1 Issue 2 January 1976

FEATURES

FEATURETTES

DEPARTMENTS

INTERFACIAL

1975 brought a giant step forward for the computer experimenter and hobbyist. It would seem that Intel and MITS started things early in the year and it's been snow-balling since.

Processor Technology, The Music Company, and Godbot have introduced the kind of competition that is making memory affordable for the affordable computer, as well as making available many of the accessory boards a hobbyist had to build a couple years ago, if he wanted one.

Other neat items like A to D converters (also affordable), PROMS pre-programmed with a working set of software, good cassette interface boards, inexpensive logic probes, and last, but not least, literature—good, readable books for the man that can't qualify as a walking sliderule.

And things are looking better every day. We are hearing rumors of floppy disk drives for under $1,000, color TV interfaces (I want one, I want one!) and new CRT terminals at around $1,000. We saw a CRT a couple of days ago that consisted of a keyboard with the CRT mounted on a column at the end of the keyboard and black box (it could be tucked away under the desk) about 6 to 8 inches square. That was it—trim, functional and anything but bulky. If that sells in the $1,000 to $1,500 dollar range, imagine what it will do to the price of those bulky and cumbersome boxes we've drooled over in the past! (Yes, brother bit busters, we've told Hal Lashlee about it; group purchase information should be available in the not-too-distant future, and we hope to have specifics with pictures, next month!)

This month we've started the hardware education campaign with an article on microprocessor theory, courtesy of Motorola. Also in the hardware department is Cliff Sparks' second installment on teleprinters, and from Scott Wilcox, some facts on grounding and findings on an A to D converter.

In the software section, for all those who bought Processor Technology's 4K memory is a routine by Jon Waldon which will thoroughly test your key to bigger and better programs. (Jon informed us the program makes approximately 208 million separate checks, and insisted on giving Ward Spaniol design credits).

You say you don't have a PT 4K board? Okay, Wilcox to the rescue—he contributed a memory check program for the Altair. And Joe Hughes contributed his favorite ASCII to binary loader.

From time to time we will be reprinting articles we feel are particularly relevant to the beginner. This month, courtesy of DIGITAL DESIGN, the novice will find a helpful article on building an Altair written by Martin Himmelfarb.

We're hoping in the months to come to steal some of the emphasis in hobby computing from games and lend it to graphics—not just characters on a CRT, but something a bit more aesthetically pleasing. To start the ball rolling, a featurette, "Culture For Computers" by John Whitney is offered to stimulate your imagination.

Your hardware gumshoe, Mike Teener, presents more of his mind-boggling findings, and Chris King in cooperation with Ralph Klestadt rounds this issue out with more on large scale systems.

To wrap thing up we present a couple of interesting letters to the editor from Larry Press and Scott Wilcox, some charts to make programming your 8080 easier, and an interesting question regarding the 8080 instruction set posed by Dick Kenyon.

You may have noticed we're a bit short on pictures. To put it another way, we need a couple of photographers and a couple of reporters. Any volunteers?

Be good to your binary beast; it might have been momma's mink. Happy switch-swatting.

Art Childs, Editor

OUR APOLOGIES

Due to circumstances beyond the editorial staff's control, some members may have received copies of the December SCCS INTERFACE missing pages 9 through 12 and pages 21 through 24.

Please inform a member of the editorial staff and a replacement copy will be sent with all possible haste.

THE COMPUTER MART

One — stop computer supermarket.

Our GRAND OPENING will be announced soon. We are still operating out of our offices and are delivering our lines of merchandise to customers. In the meantime we will serve you out of the office.

The first deliveries of the IMSAI 8080 are being made. Acceptance is great.

AUTHORIZED DEALER FOR:

- IMSAI Computers and peripherals.
- PROCESSOR TECHNOLOGY
- CROMENCO — The TV DAZZLER and the 8KBYTESAVER®
- SOLID STATE MUSIC — Memories and Boards.
- VECTOR ELECTRONICS — Kit builder supplies and parts.

Inventory from these suppliers and manufacturers is being stocked in order to give you off the shelf delivery.

Processor Technology

What will you like about our static 4K memory ? ?

THE PRICE ! ! $139.00

SPECIFICATIONS: 4KRA-4
Maximum capacity: 4096 eight bit bytes
Operating mode: Static
Access time: 520 nano-seconds, worst case maximum
Cycle time: 520 nano-seconds maximum, read or write
Bus Pinout: Plug-in compatible with Altair 8800 Bus
Edge contacts: Gold plated, 100 pins (dual 50) on .125" centers
Power requirements, operating: +7.5 to +10VDC at 1.0A maximum (0°C), 0.8A typical at 25°C.
standby: +1.6 to +2.5VDC at 0.5A maximum worst case, 0.4A typical
Dimensions: 5.3" x 10.0" (13.46cm x 25.4cm)

THE COMPUTER MART
Authorized Dealer

2333 Beverly Boulevard • Los Angeles, CA. 90057 • (213) 484-2002

UPdate

JANUARY MEETING — January 18, 1976 — Sunday
TRW Systems, Redondo Beach
Building "S" Cafeteria

Doors open 10:00 a.m. Coffee hour is 12:00-1:00 p.m.

Meeting 1:00-4:30 p.m.

FEBRUARY MEETING — Saturday, the 21st, same place at TRW

NOVEMBER MEETING The November meeting was unusually interesting. Mr. Ted Wirtz gave an informative talk and showed a short but fascinating film on bubble memory. As a result of Mr. Wirtz's skillful sharing of his knowledge, visions of sugar plumbs were replaced in many computer buff's Christmas dreams this year.

The November meeting was also notable for the appearance of several new manufacturers on the computer hobbyist scene. Among the equipment displayed was a fine looking 8080 based computer, and an Altair compatible 6800 CPU card.

Several members walked away from the November meeting a bit wealthier than when they arrived as a result of the drawing. Edwards Associates donated a set of bug books, Intel donated a set of 8080 books, the club contributed a couple of pocket calculators and a nice pen. Current plans call for the drawing to be a regular event. Donations for the December meeting were made by E & L: a bread board kit, The Computer Store: some computer lib books.

DISCOUNT ON BUG BOOKS

Those readers familiar with the Bug Books will be happy to learn of the discount given by Edwards Associates on the purchase of Bug Books when purchased at the monthly meetings.

TRW SYSTEMS, By their consent to the use of the Building S Cafeteria for the monthly meetings, has been instrumental in making the Southern California Computer Society the success it has so obviously become. Attendance at the November meeting was close to 500, a number that would be prohibitively expensive to accommodate by a non-profit organization charging dues of only $10 a year. To the Officers of TRW—our heart-felt thanks for their generous assistance.

MICROCOMPUTER DESIGN AND CONSTRUCTION CLASS

THE FIRST HARDWARE CLASS

The first hardware construction class has commenced. The response to the letter (printed below) was immediate and positive, and has resulted in tentative plans for more classes.

Here is a chance for four "software types" to become "hardware types"! Pat Amornvitikivcha will run a workshop/class which will design and construct an 8080 based microcomputer.

The system will be similar to one which Pat has built for himself. It will be interface compatible with ALTAIR I/O boards, memory, etc.; however, it will not be an exact copy of an ALTAIR. For example, the front panel will feature keyboard input, rather than switches, and direct octal display.

Pat estimates that the project will require no more than 40 hours total time and wants to meet for 5 to 10 sessions of 4 to 8 hours each. The classes will be arranged for weekends and evenings. The class will be held in Venice and there are no prerequisites. Tuition will be $25 and the enrollment will be limited to 4 people. If you are interested contact Pat at (213) 384-8159 or write 828½ S. Berendo St., LA 90006.

The class tuition will be used to buy parts (the SCCS treasury will match the tuition) and the computer will belong to SCCS when the project is finished. The goal here is to provide some in-depth training for members while getting SCCS some hardware as a by-product. We hope that this is just the first of a series of construction/design projects. If you would be interested in leading a (hardware or software) project or have some ideas as to projects that you would like to see offered, contact Larry Press at (213) 399-2083 or write 128 Park Place, Venice, 90291.

PROGRAM TOPIC FOR JANUARY MEETING

Don Tarbell, Chairman of the Program Committee, informs us the speakers at the January meeting will be Mr. John Titus, co-author of "The Bug Books", speaking on the 8080 microprocessor, and Jerry Silver, whose topic will be Introduction to Software.

Planned for topics at the February meeting are Floppy Disk Systems, presented by Bob Baskin of ICOM, INC., and Microprocessor Development Center, presented by Hamilton/Avnet.

ATTENTION SCCS MEETING EXHIBITORS

Due to the limited space available at the facility being used for SCCS meetings, and the attendant cleanup problems, the number of exhibitors that can be accommodated at future meetings must be restricted.

Persons wishing to exhibit are requested to contact:

Ron Keele
9823 Redfern Ave.
Inglewood, Calif.
Phone (213) 672-5437

Allocations of space will be made on a first come, first serve basis.

CLUB FACILITY PROGRESS REPORT

Mike Dent of the Facilities Committee poses the following question:

"Would you like to be able to assemble your pet project without cluttering up the kitchen—without the fear of curious jam-covered little hands eagerly examining your recently arrived front panel—a place where there is available a scope and a DVM for use in determining why your memory board doesn't remember?"

Mike reports, "The SCCS wants to provide such a place. The Facilities Committee is planning to open a workshop with tools, test equipment, supplies and people who know how to make computers work."

The committee would like to offer all this at little or no cost to club members. To do this, the club needs donations or loans of equipment, tools and books, as well as time from members with hardware skills.

Some progress has been made in this direction. As a result of the last Board of Directors meeting, two sites are currently under investigation. The selected site will eventually have a hardware workshop, a classroom/meeting room, a library, a terminal connected to a time share system, a club office, and hopefully, a coffee pot.

Art Childs, being too busy as Editor of SCCS INTERFACE to use his TV typewriter, has loaned it to the club for use as part of the terminal.

Also needed are lights, work benches, shelves, chairs, a desk or two and probably paint and lots of elbow grease.

This request for assistance could be an opportunity for the various manufacturers and suppliers to gain exposure for their equipment and to give a boost to the computer hobbyist movement (some still think the Altair is the only affordable computer on the market).

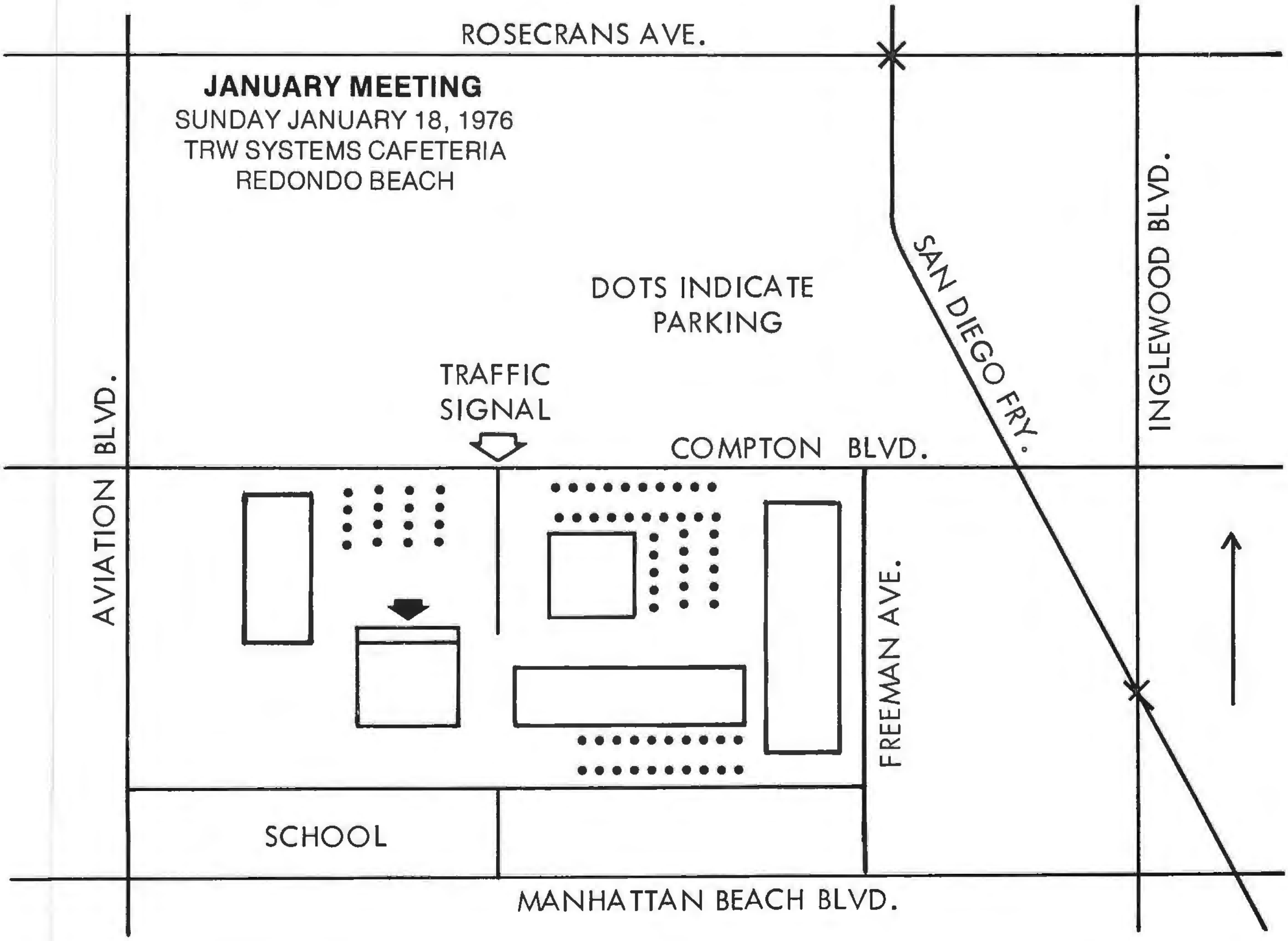
Those with donations or loans to this fine cause are urged to write Mike Dent, P.O. Box 4581, Inglewood, Calif. 90309, or call him at (213) 677-5433.

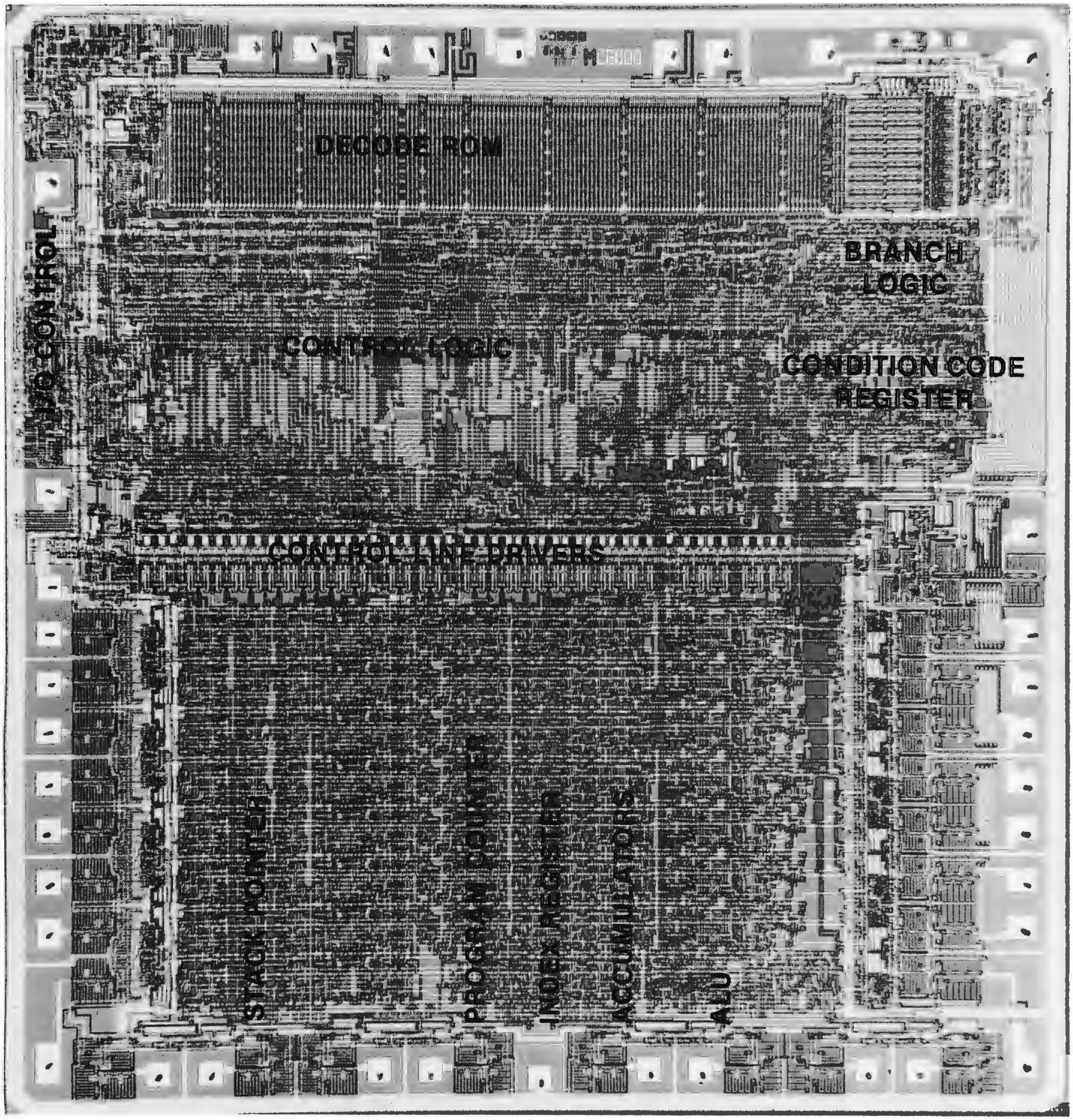
VENTURA COUNTY CHAPTER FORMING

The first meeting of computer hobbyists and experimenters in Ventura County is planned for January 24, 1976. The meeting place has yet to be determined, but interested persons may call either of the two people listed below for further information.

John Borders (805) 982-5685 or (805) 985-1631
Eric Strohbehn (805) 982-8538

Good luck and welcome to the Southern California Computer Society. ◘





MC6800 Micro Processor Chip

Photo courtesy of Motorola Semiconductors, Phoenix, Arizona

INTRODUCTION TO MICROPROCESSOR TECHNOLOGY

From "Introduction to Microprocessors" reprint courtesy of Motorola Semiconductor, Phoenix, Arizona.

Interface has serialized this new and very informative book and will be presenting it in its entirety over the next several issues.

Chapter I

WHAT IS A COMPUTER?

1. Types of computer

A computer is a machine capable of carrying out arithmetic or logic operations on data which are presented to it at the input and of providing numerical results or *decisions* at the output.
Three types of computer may be distinguished:

- *Analog computers*
  The information is in analog form and is processed in this form (a magnitude is represented by a voltage)
- *Digital computers*
  The information is in binary form (series of 0's and 1's)
- *Hybrid computers*
  These combine the characteristics of the two previous types.

2. Digital computers

A digital computer is capable of *storing* and *processing* information in digital form. However, this first definition is not sufficient because it would imply that a desk calculator was a member of the computer family. This is incorrect because there is an essential difference between the computer and the calculator: the first is capable of carrying out any operation whereas the second can only perform the function that was permanently assigned to it when it was built.

This ability of the computer to accomplish a variety of tasks dictated by a series of instructions has a counterpart: the need to programme it and therefore to know the language which the computer can accept.

This distinction between the computer and the calculator now enables us to give the definition which appears most accurate.

3. Definition

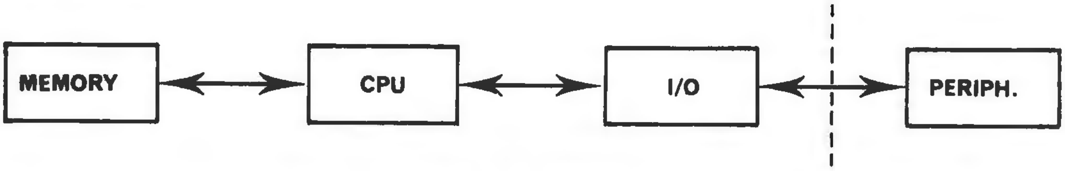
Any hardware system will be designated "computer" as opposed to calculator when the following conditions are fulfilled:
– it has a random access memory for read/write operations;
– it has a controllable input-output system;
– its repertory of instructions allows:
a) the manipulation of words stored in the memory (arithmetic, logic or transfer operations)
b) the modification of any bit in a word;
c) transferring the control of a programme by branching when the necessity arises (decision making power of a computer)
d) controlling the external equipment with the aid of an interruption facility;
– the instructions, that is the programme, are stored and processed using the same hardware as for the data.

This definition already allows us to get a first idea of the structure of a computer or of a mini-computer, which has the same organization as a big computer but differs from it essentially as regards price, performance and the field of applications which it covers.

Chapter II
THE ORGANIZATION OF A COMPUTER

A. GENERAL

1. Block diagram and function of the main parts



The memory of a computer contains a certain number of cells in which the information can be either an item of *data* to be manipulated by arithmetical or logical operations or an *instruction* which specifies the operation to be carried out.

The CPU (Central Processing Unit) is a unit equipped with an arithmetic and logic unit capable of carrying out the operation specified by the instruction and a control unit capable of controlling the sequence of different steps required by the instruction to be processed. The operation of the computer implies that the instructions are stored in the memory in a sequential fashion, that is, one after the other. Establishing the series of instructions or "programme" which must be provided to the computer for it to carry out a task is called "programming".

In order to be able to provide the computer with the programme and the data to be manipulated, it is necessary to establish a means of communication between the computer and outside. This is the function of the Input/Output (I/O) units which, whilst providing this link with the peripheral units, also have to convert the external data into a form usable by the computer (interface function).

Examples:

a) the CPU-teletype interface transforms series data into parallel data;

b) an analogue digital converter forms the interface between the CPU and a thermocouple; etc.

To summarize:

a) the memory is a storage unit

b) the CPU performs calculation and control functions

c) the I/O units permit communication of the CPU with the outside.

2. Analogy

The operation of a computer may be compared with the thought processes in man when he wants to add up several numbers.

The sheet of paper on which are noted the numbers and on which the result is written represents the peripheral unit, the source and the destination of the data to be processed.

The eyes and the hand constitute the I/O system, the eyes enabling the numbers to be put into the memory and the hand enabling the results to be written down. The two memories, that of the machine and that of man, both have the same function, both contain the addition algorithm which the man learns as a child.

The part of the brain which carries out the addition while co-ordinating the different operations represents the CPU. The comparison between man and computer stops here, unfortunately for us who do not have the faculty of effortlessly carrying out a repetitive task at high speed as can the computer. On the other hand, the computer's lack of intuition and imagination and its need for explicit instructions on every detail of what it must do, makes it much inferior to man.

B. DESCRIPTION OF THE SUB–SYSTEMS

1. The memory

1.1. *General*

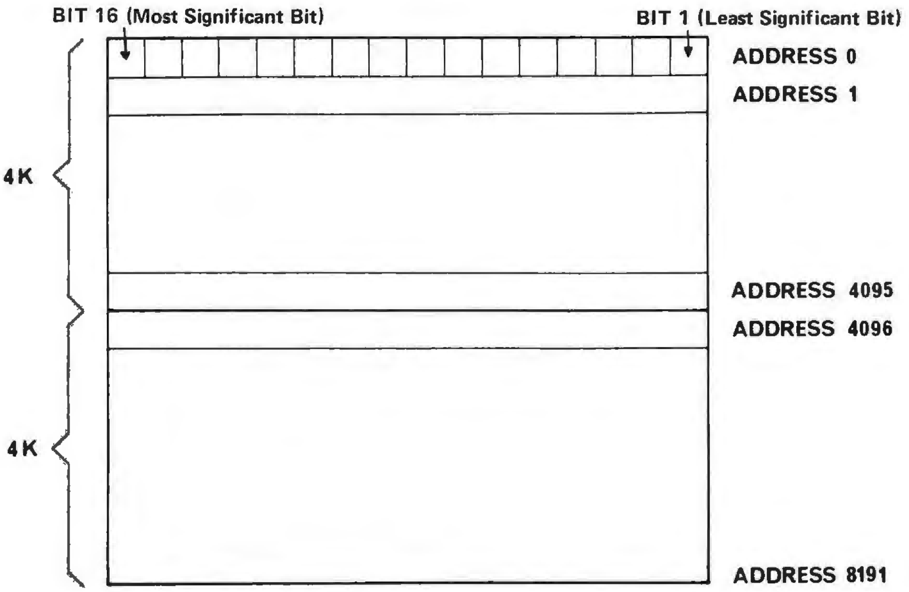
This is the unit that stores the *bits* (*b*inary dig*its*) of instructions and data in a large number of cells or locations. The information stored in groups of cells constitute a *word* of n bits. A word is comprised of one or more bytes; a byte is defined as a group of bits (4, 8, 16, etc.) that are processed as a single item.

The precision of a calculation is dependent on the number of bits used to represent a number and therefore on the size of the words. The size of the words varies from one machine to another; for a minicomputer it ranges from 8 to 24 bits. The size of the memory is equal to the number of locations it contains. In general it is a multiple of 4 K words (1 K word = 1024 words of n bits) because a modular design is nearly always adopted.

Each location of the memory is identified by a unique *address* which allows access to the contents of the location. Consequently, to obtain an item of information from the memory it is necessary to know the address at which this information is stored.

At any given time it is only possible to address a single location.

A memory of 8 K 16-bit words may be represented as follows:



1.2. *Types of memory*

Semi-conductor memories
Magnetic memories
- core memory
- drum memory
- disc memory
- magnetic tapes

1.3. *Communication between memory and CPU*

This is achieved by means of two buses and two registers:
- Address Bus and Address Register for the addresses
- Data Bus and Data Register for the data, that is, for the *contents* of the memory locations specified by the address.

When the CPU wants to read the information in a given location, it puts the address of the location in one of its internal registers, the Address Register, and then transfers the contents of this register AR to the memory on the Address Bus. After decoding the address, the location selected sends its contents to the CPU on the Data Bus. These contents are stored in another register of the CPU termed Data Register.

The same principle applies to writing: the contents of the Data Register are stored at the address indicated by the Address Register. The mode of access to a memory position is termed *random* because it allows immediate access to any memory position; in contrast, there also exist memories with *sequential* access (magnetic discs).

A memory which permits random access is termed a *random access memory* (RAM) or *read/write memory.*

Certain memories only allow reading; the information is frozen in and can not be destroyed by overwriting. These are *fixed memories* or *read only memories.*

The memory of a mini-computer can be composed of both ROM's and RAM's together. In the ROM's may be stored the permanent utility programmes such as the programmes for reading a punched tape from a high speed or slow reader. Constant values may also be stored here while the RAM's will contain the programme to be executed and the variables of the problem.

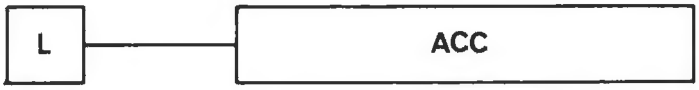
Example: Calculation of the pay of an employee; the rates for an hour of normal work and for an hour of overtime can be stored in the ROM while the variable values which are the number of hours of each type done by the employee will be stored in the RAM .

2. The processor or CPU

This is the active part of computer and is composed of several sub-systems of which the most important are the following:

The Accumulator Register ACC in which take place the arithmetic and logic operations.

The Link or Carry Register or flip-flop. This is a register of 1 bit which is considered as an extension of the accumulator; it is used in particular to connect via a loop the MSB and the LSB of a number contained in the accumulator during rotation operations.



The Status Register which is composed of 5 special flip-flops:

a) the Carry flip-flop just mentioned which is affected either by rotation operations or by arithmetic or logical operations giving rise to a carry;
b) the Negative flip-flop, set to 1 when the contents of the accumulator become negative;
c) the Zero flip-flop, set to 1 when the contents of the accumulator become zero;
d) the Overflow flip-flop, set to 1 when a operation causes overflow on the sign bit of a number;
e) the Interrupt flip-flop which permits a request for interruption to be granted or not.

The Instruction Decoder register (I.D.) in which is stored the operation code part of an instruction; in general, an instruction is composed of 16 bits, for example, the 4 most significant bits could indicate the operation code such as that for addition and the other bits the address fo the operand.

The Programme Counter register (PC)

The computer memory contains the programme to be followed, that is, a series of instructions. Consequently, at any time it must know which instruction is to be carried out or rather at which address the instruction is stored in the memory. The role of the PC is to hold this address. At the end of each cycle, the PC indicates the address of the next instruction to be carried out; this thus allows the sequential execution of a programme of which the instructions are also sequentially stored in the memory.

However, in some cases, the contents of the PC may be modified by the programme itself. In this way it is possible to carry out instructions stored in another part of the programme. This is done by branching instructions.

The control unit

This unit co-ordinates all the parts of the computer in such a way that the events take place according to the correct sequence and at the right time, that is, this unit supervises the correct execution of each cycle, the number of cycles depending on the instruction to be processed (see chapter III).

3. The I/O system

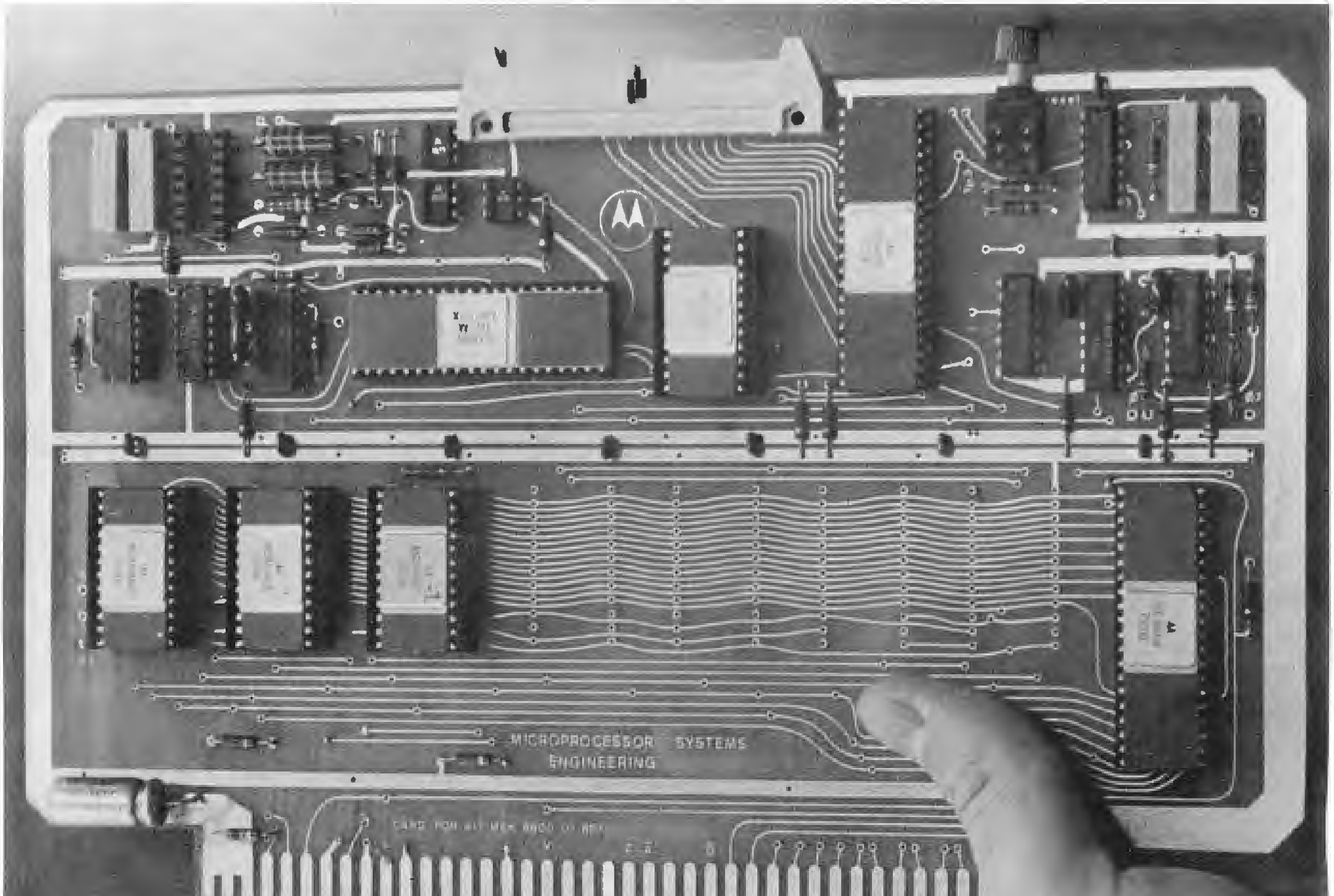
This provides the interface between the Processor and the outside (peripheral unit). To do this, it must be able to solve:

- The problem of timing: peripheral equipment is generally much slower than a computer.
- The problem of the format of the information transmitted to the computer (series-parallel translation).
- The problem of hardware: a peripheral unit does not necessarily have the same logic as the computer.

Criteria for the appraisal of a computer

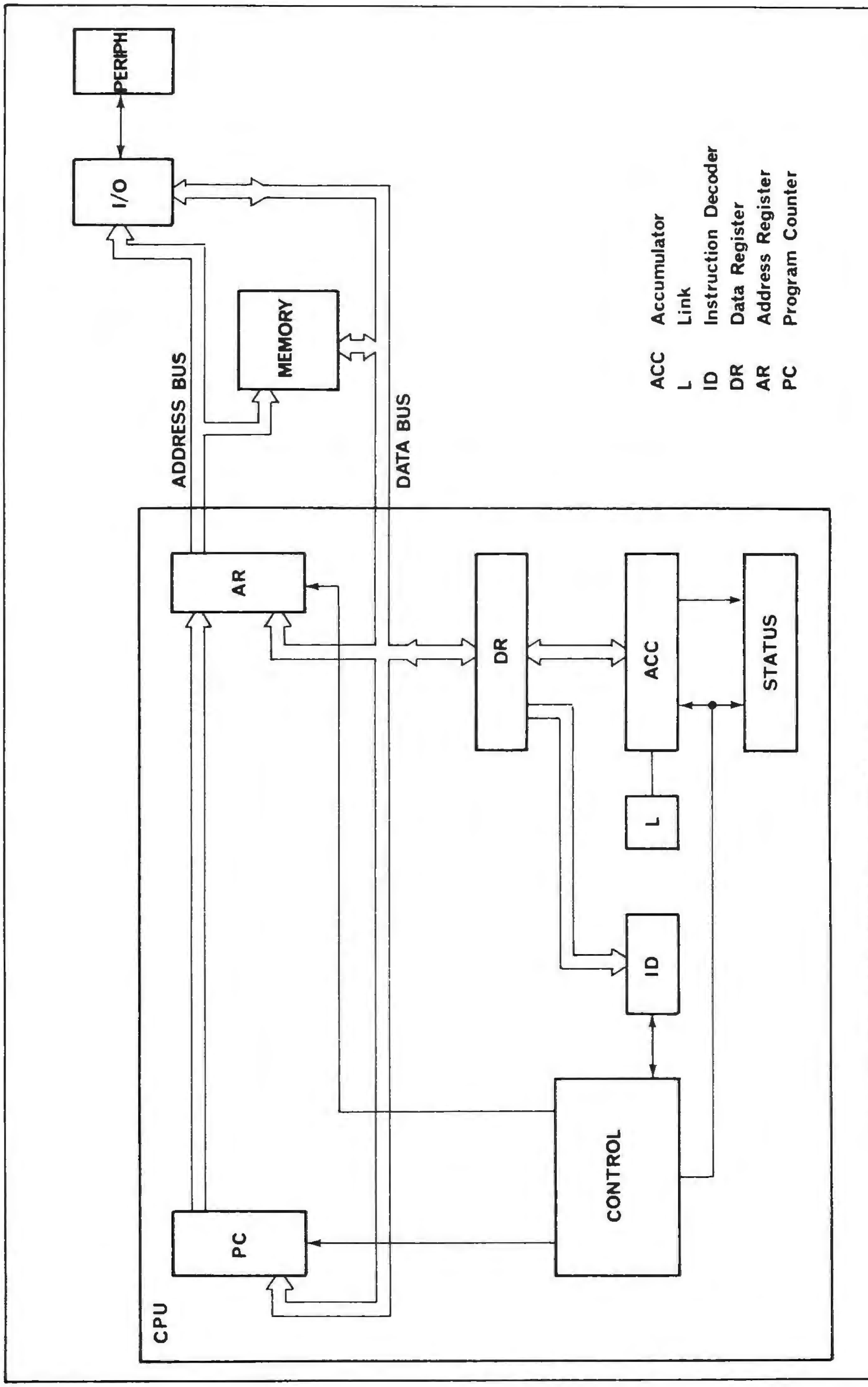
The three main parameters which characterize a computer system are:

- the size of the memory
- the speed of execution of the instructions
- the number and type of instructions available.



Motorola MEK 6800 D1 Microcomputer Kit

Photo courtesy of Motorola Semiconductors, Phoenix, Arizona



TYPICAL INTERNAL ORGANIZATION OF A CPU

Chapter III

DESCRIPTION OF AN ELEMENTARY OPERATION –CYCLES

The aim of this chapter is to explain the operation of the Processor by means of a simple example which we will break down into the different steps.

The example chosen is that of the addition of two numbers stored in the memory at addresses A and B; the result, number 3, of the addition of these two numbers 1 and 2 will be stored at the address C.

This job requires four instructions:

a) an instruction enabling the ACC to be loaded: LOAD MEMORY. The contents of the memory address are transferred to the accumulator ACC;

b) an arithmetic instruction effecting the addition in the accumulator ACC; this is the instruction ADD MEMORY which adds the number stored at the memory address B to the contents of the accumulator, the result automatically remaining in the accumulator;

c) an instruction enabling the contents of ACC to be stored: STORE MEMORY. The contents of ACC are unloaded at the memory address C;

d) a control instruction STOP which stops the execution of the programme.

The programme will consist in loading ACC with the contents of the address A (LOAD A), adding the contents of the address B (ADD B), storing the result at the address C (STORE C) and stopping the work (STOP).

(Loading could have been done by means of switches.)

Address	*Contents*
A	Number 1
B	Number 2
C	Number 3
D	LOAD A
D+ 1	ADD B
D+ 2	STORE C
D+ 3	STOP

Note:

The three locations reserved for the data may be anywhere in the memory but the positions occupied by the programme (instructions) must be sequential.

To carry out this programme it suffices to:

– set the PC to the initial value D (1st instruction)

– request execution (actuate the "RUN" switch of the computer).

Let us analyse what happens when these two requirements are fulfilled, adopting the following conventions:

(X) indicates the contents of the memory position of address X or the contents of the register X

$(X)_1$ indicates the operation code part of an instruction

$(X)_2$ indicates the address of the operand which is involved in the instruction.

The following steps will take place sequentially:

– *FETCH cycle, instruction LOAD A*

a) (PC) = is transferred to the address register AR

b) the PC is incremented by 1: PC + 1→PC;

c) the contents of the address selected, i. e., (D) are transferred to the data register DR;

d) the operation code part of DR, i. e., $(DR)_1$ is sent to the instruction decoder register I. D.;

– *Execute cycle, instruction LOAD A*

a) the operand part of the word contained in DR, i., e., $(DR)_2$, which contains the value of the address A, is transferred to AR;

b) the contents of the position addressed by AR, i.e., (A) = Number 1, are transferred to DR;

c) (DR) is transferred to the accumulator ACC by the hardware circuits of the CPU which have been activated to carry out an ACC loading operation because ID has decoded the instruction LOAD

After execution of these *two* cycles corresponding to the *single* instruction LOAD A, the position is as follows:

(ACC) = Number 1

(PC) = D + 1 (The PC was incremented at step 1b).

– *FETCH cycle, instruction ADD B*

a) (PC) →AR so that (AR) = D + 1

b) PC + 1 →PC so that (PC) = D + 2

c) (D + 1) →DR so that DR contains the machine code of the instruction ADD B

d) $(DR)_1$ →ID so that ID contains the operation code ADD

– *Execute cycle, instruction ADD B*

a) $(DR)_2$→AR so that AR contains the value B which is the address of number 2

b) (B) = Number 2→DR

c) The appropriate circuits add the current contents of DR to the contents of ACC where the number 1 has been stored since step 2c. The result remains in ACC.

Thus, we now have:

(ACC) = Number 1 + Number 2

(PC) = D + 2 (since step 3b).

– *FETCH cycle, instruction STORE C*

a) (PC)→AR so that AR = D + 2

b) PC + 1→PC so that PC = D + 3

c) (AR) = (D + 2)→DR which contains the machine code of the instruction STORE C

d) $(DR)_1$→ID which holds the operation code STORE

– *Execute cycle, instruction STORE C*

a) $(DR)_2$ = C→AR

b) (ACC)→DR

c) (DR) = Number 1 + Number 2 is stored at the address indicated by AR

This is a WRITE operation as the CPU is putting an item of data in the memory.

In contrast the steps 2b and 4b, where the CPU reads the data in the memory, are termed READ operations.

We have now stored the result of the addition in the memory and the PC currently indicates the address $D + 3$.

– *FETCH cycle, instruction STOP*

a) $(PC) = D + 3 \rightarrow AR$
b) $PC + 1 \rightarrow PC$
c) $(AR) = (+ 3) \rightarrow DR$ which contains the machine code of the instruction STOP
d) $(DR)_1 \rightarrow ID$

– *Execute cycle, instruction STOP*

In ID the control unit has decoded a STOP instruction. All activity of the CPU is therefore suspended. The instruction at the address indicated by the PC, i.e., at the address $D + 4$, will not be fetched.

This intentionally simple example was mainly intended to demonstrate the two principal cycles necessary for processing an instruction. More complex cycles may be added to these two basic cycles.

(Continued on page 53)

For a limited time only, you can own an Altair® 8800 Computer kit with 4,096 words of memory, new Altair multi-port interface, and revolutionary Altair BASIC language software, for just $695. A savings of up to $114!*

Computer. The Altair 8800 is the best-selling general-purpose computer in the world today. It is a parallel 8-bit word/16-bit address computer with an instruction cycle time of 2 microseconds. It was designed for almost unlimited peripheral and memory expansion, using a bus system where all input/output connections merge into a common line. The Altair 8800 is capable of addressing up to 65,536 words (bytes) of memory. *Regularly priced at $439 for a kit and $621 assembled.*

Memory. The Altair 4K Memory Board provides 4,096 words of dynamic random-access-memory for the Altair 8800. Contains memory protect circuitry, and address selection circuitry for any one of 16 starting address locations in increments of 4K. Access time is 200-300 nanoseconds. The entire 4,096 words of memory on the board can be protected by switching to PROTECT. *Regularly priced at $195 for kit and $275 assembled.*

Interface. Your choice—either the *new Altair 88-2SIO serial interface* or the *new Altair 88-4PIO parallel interface.* The serial interface can be ordered with either one or two ports and the parallel interface can be ordered with up to four ports. *Add $24 for an additional 88-2SIO port kit. Add $30 for each additional 88-4PIO port kit.*

Each port of the *new serial interface board* is user-selectable for RS232, TTL, or 20 milliamp current loop (Teletype). The 88-2SIO with two ports can interface two serial I/O devices, each running at a different baud rate and each using a different electrical interconnect. For example, the 88-2SIO could be interfaced to an RS232 CRT terminal running at 9600 baud and a Teletype running at 110 baud. An on-board, crystal-controlled clock allows each port to be set for one of 12 baud rates. *The 88-2SIO is regularly priced at $115 kit and $144 assembled.*

Each port of the *new parallel interface board* provides 16 data lines and four controllable interrupt lines. Each of the data lines can be used as an input or output so that a single port can interface a terminal requiring 8 lines in and 8 lines out. All data lines are TTL compatible. *The 88-4PIO regularly sells for $86 kit and $112 assembled.*

Software. Altair 4K BASIC leaves approximately 725 bytes in a 4K Altair for programming which can be increased by deleting the math functions (SIN, SQR, RND). This powerful BASIC has 16 statements (IF . . . THEN, GOTO, GOSUB, RETURN, FOR, NEXT, READ, INPUT, END, DATA, LET, DIM, REM, RESTORE, PRINT, and STOP) in addition to 4 commands (LIST, RUN, CLEAR, NEW) and 6 functions (RND, SQR, SIN, ABS, INT, TAB, and SGN). Other features include: direct execution of any statement except INPUT: an "@" symbol that deletes a whole line and a "←" that deletes the last character; two-character error code and line number printed when error occurs; Control C which is used to interrupt a program; maximum line number of 65,529; and all results calculated to seven decimal digits of precision. *Altair 4K BASIC is regularly priced at $60 for purchasers of an Altair 8800, 4K of Altair memory, and an Altair I/O board. Please specify paper tape or cassette tape when ordering.*

***Savings depends upon which interface board you choose. An Altair 4K BASIC language system kit with an 88-2SIO interface regularly sells for $809. With an 88-4PIO interface, this system sells for $780.**

NOTE: Offer expires on March 30, 1976.

MITS

"Creative Electronics"

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MAIL THIS COUPON TODAY!

□ Enclosed is check for $____________________
□ BankAmericard #__________ □ or Master Charge #__________
□ Altair BASIC System Special □ 4PIO interface □ 2SIO interface
□ Cassette tape □ or paper tape
□ Extra SIO port □ _____ Extra 4PIO ports
Add $8 for postage and handling.
□ Please send free literature

NAME________________________________

ADDRESS______________________________

CITY____________________ STATE & ZIP__________

NOTE: Personal checks take 2-3 weeks for clearance. For immediate processing send money order or use charge card. Delivery: 30 days. Prices, specifications and delivery subject to change.

The IBM System/370 Model 158, large-scale computer

TOWARD THE UNDERSTANDING OF LARGE SCALE SYSTEMS

by Ralph Klestadt and Chris King

EDITOR'S COMMENT: The Southern California Computer Society is fortunate to have among its members two unusually intelligent and knowledgeable students: Chris King and Ralph Klestadt. Although their interest is in systems which are much larger than those the average computer hobbyist will experience, their knowledge is certainly worth sharing.

The series of articles written by these two gentlemen is being published in the hopes it will stimulate the interest of other students, as well as for general interest.

Last issue we defined what a large scale system is, but we have yet to examine the different types of systems. Since many types of systems are situated for different applications, let's see what system is best for what application.

We will be looking at the two major types of systems—true batch and true timesharing. We say "true" because many manufacturers advertise their machine as having timesharing, but in reality it is just "very fast" batch. True timesharing is the actual machine action of timesharing.

BATCH

A batch system is situated for a sequential and fast processing environment. The portions of the batch system which are visible to the user's job are queue, the CPU, and the I/O devices. Let's look at each of these and see how job flow is handled.

The Queue

Once the programmer has punched up his program onto cards, this is submitted to the queue by means of a card reader. Once the reading is complete, an exact duplicate of the program on cards is created on a magnetic holding device (usually disk), called the queue. Here jobs are held and assigned priorities (depending on your account number, job parameters, etc.) for processing. The CPU will be constantly taking the job in position one. If your job gets in as job 295, as the CPU takes each job 1, your job will come down in the queue, to 294, 293, and so on until you are job 1, whereupon your job begins processing. However, if a job gets entered right after yours and it has a high priority, it will be executed before yours. Being "held in the "queue" is when your job has been entered and either some of the jobs before you are taking a long time in the CPU or your job has a low priority and is moved to the end of the queue. You may wonder how being job 295 wouldn't take a long time to get executed. Well, batch machines are especially situated for this environment, so they will execute jobs very quickly.

THE CPU

The CPU, or Central Processing Unit, is where the program is executed and where it gains its control over devices (that it has access to). Once in the CPU state of execution, the program is in a state of execution where it will begin giving input to the Monitor. The Monitor is a very large program which has control over all functions of the system. Normally, the user program can not get control of the Monitor, because if it could, he could foul up the system quite easily. Once in Monitor state, the user program will usually have job control statements which tell the monitor what to do, such as call FORTRAN and run the program in FORTRAN, call a utility (special purpose system program, for example to generate reports) and use the following cards as data, or perhaps copy a file from a magtape to file on a disk.

The CPU will normally consist of the ALU (Arithmetic Logic Unit), some memory (usually over 100K on a large scale system), and some channels for I/O. Normally, the user program will have access to most devices, unless he has very low priorities or parameters that are invalid.

THE I/O DEVICES

The purpose of this section is not to tell about the differenct I/O (Input/Output) devices, but rather to tell about flow of programs concerning them.

After or during execution of the job, the job will have to exchange with or give data to sources other than the program, such as printing the output onto a printer. Devices most visible to the user's job will be mainly the printer, disk units, and tape units. When the user calls for something to be printed, the output to be printed is often handed over to a channel, which can be compared to a mini- or micro-computer. This takes the output and assembles it in proper format, then prints it out (as the case may be . . . on a disk it would write it rather than print it.) Input from disk or tape depends largely on machine or language used, but will usually slow down execution time. In the end, output from the program may be printed, or magnetically recorded or both. Often punch cards for output are used, but this is no longer common practice (from what we've seen).

Note: The system we've used for examples in the above section was the IBM/370 line, and some statements above may not be true for other machines.

TIMESHARING

A timesharing configuration is situated mainly for the user who wants an interactive system, or a system where he gets immediate answers to his questions and where things can be developed and changed easily Although timesharing is more expensive than a batch system, it allows several users (up to 512 on the DEC System-10) to be using the system all at the same time. All users have access to the same programs from their remote Teletype® or related terminals. Since all operations in timesharing are combined (no queue or I/O channels) we will just have one section about the CPU, or as it is often called, the Processor.

THE PROCESSOR

You may wonder how one processor can handle 512 jobs all at once. Well, it does it like this: when each user logs on to the system from his port, he has a small portion of processor area for his program. Even though he has little processor area, this is all he will need, because the processor will do a "running execute". This process continues until the processor gets back to our friend's job, at which time it executes the next few instructions, and so on until it finishes all jobs.

The timesharing processor is so fast that each user appears to have exclusive use of the whole computer. Actually, each job is being done in little bits (pardon the pun). Even though you have only a small execution area in the Processor, you will have available all the core in the machine needed by your program. To make things a little clearer about how the processor does its work, let's assume you log on as job 12. Your program will be executed 12th, but you have no significant advantage over the execution time of job 1 or job 40 because all jobs are executed in little bits so fast.

From his terminal, or port, the user may run system programs such as utilities, mathematical or business programs, or if he wants, games. He may, if he is allowed, access another user's programs which have been saved on disk (called the Directory), for his use. No job control statements are required in timesharing.

The big advantages of timesharing (advantages that you pay for!) are: you can interact with your program, changing values in your execution from previously found values quickly and without having to punch, submit, and wait for your output on a batch system. The other very nice feature of a timesharing processor is the text editor programs, which provide a quick, easy, and cheap way to make major alterations to your programs quickly. Needless to say, batch does not have this feature.

I/O in your timesharing job can be directed to and from disk, terminal, tape, or many other devices. One other nice feature of timesharing I/O is it defaults to terminal and disk for things which, in the batch environment, the user would normally have to specify many parameters.

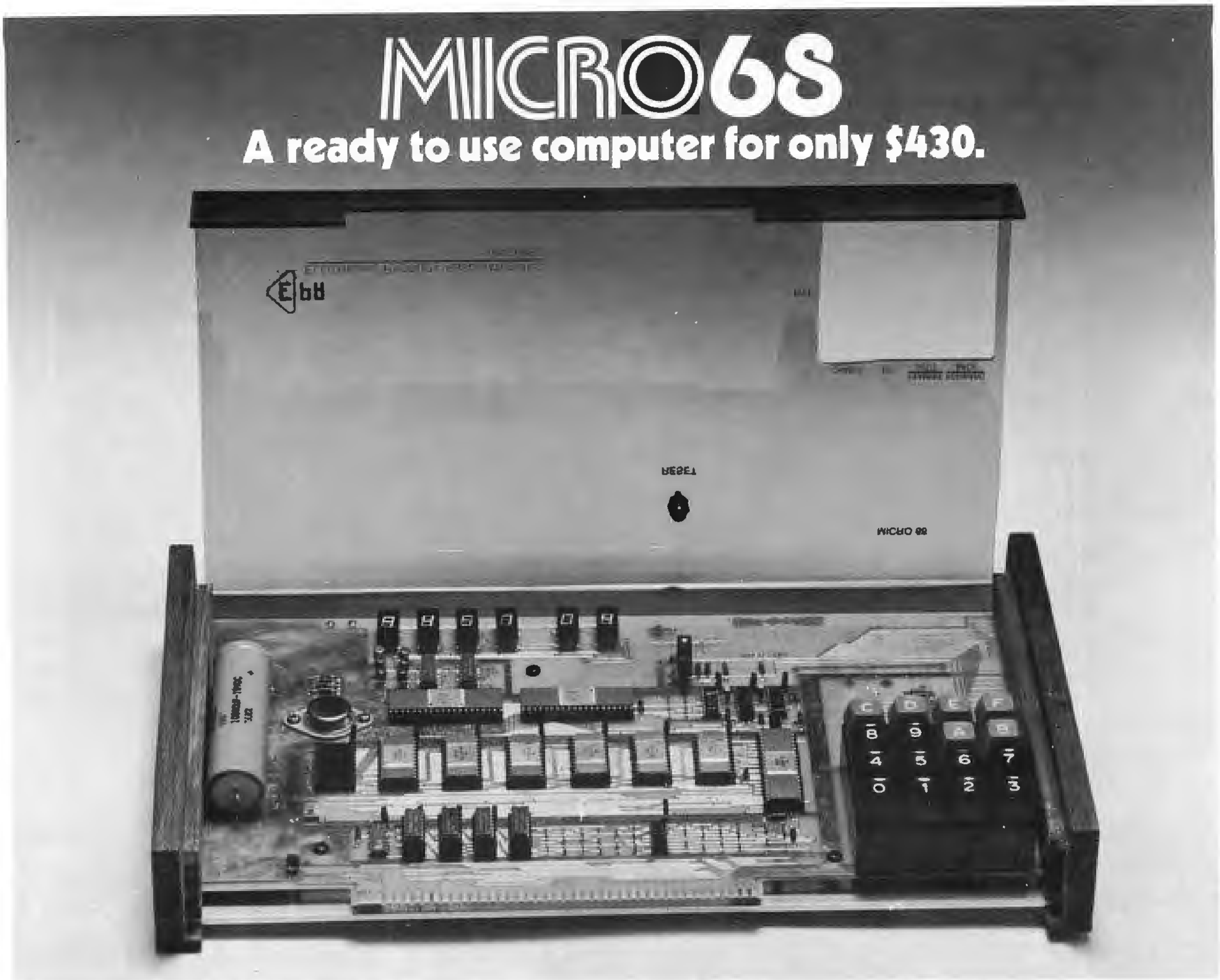
Note: The system we used for examples in the above section was the Digital Equipment Corporation DECsystem-10 line, and some statements above may not be true for other systems.

Do not look at a batch system as being useless as compared to the timesharing system. Quite the contrary. The two types of systems should be used for different purposes (note the word "should"; they would be best used that way, but are not always). A batch system should most commonly be used for production type program runs and for expensive and easy-to-develop programs, whereas a timesharing system is best to use if the user wants a system he can interact with and get the problem solved quickly and cheaply. A timesharing system can be used for production runs but will not be quite as efficient as a batch machine, whereas a batch machine can be used to develop programs but is not that easy to interact with through cards and, likewise, debug through cards. Even though the batch CPU is faster than the timesharing processor, it does not get as much accomplished in the same time period as the timesharing processor does.

On more interesting note: the IBM/370 offers timesharing ("fake" timesharing) which is not as widely used as their batch, and the DECsystem-10 offers fake batch (the cards handled like a timesharing job) which is very easy, but is not as widely used as their timesharing system.

Timesharing, batch processing, remote batch, and real-time computer operations can be performed concurrently by the DECsystem-1080 large-scale computer system.

Photo courtesy of Digital Equipment Corporation, Marlboro, Mass.



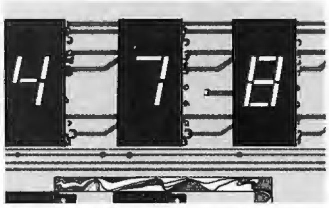
MICRO68

A ready to use computer for only $430.

The new breed of microprocessor.

The Micro-68 is a complete computer system built around the famous Motorola/AMI 6800 Microprocessor. The Micro-68 comes complete and ready to run, with its own integral power supply, keyboard and display. The 512 word John-Bug Prom contains all the service programs necessary to load your own programs, inspect and edit them as necessary, insert break points for debugging purposes and execute. Convenient edge connectors are provided for memory expansion to 64K and full 16 Bit I/O. All memory lines can be fully buffered on-board.

The Micro-68 also features a 16 button keyboard, a 6 digit LED display, 128 words of RAM, its own power supply and sockets for 768 words RAM on board. Never before has so much capability been offered at such a low price. The action is here and now, with the only industry-wide standard Microprocessor — the 6800. Electronic Product Associates, backed by 18 years of experience, provides you with guaranteed quality in both design and construction. The revolution has begun.



EPA **Electronic Product Associates, Inc.**
1157 Vega Street
San Diego, California 92110
(714) 276-8911

MICRO68
Now available from
WEATHERFORD

Albuquerque: (505) 265-5671
Anaheim: (714) 634-9600
Dallas: (214) 243-1571
Denver: (303) 761-5432
Glendale: (213) 849-3451
Houston: (713) 688-7406
Palo Alto: (415) 493-5373
Phoenix: (602) 272-7144
Pomona: (714) 623-1261
San Diego: (714) 278-7400
Seattle: (206) 243-6340

ASCII TO OCTAL

MACHINE LANGUAGE LOADER

by Joe Hughes

EDITOR'S NOTE: In a phone conversation with your editor, Joe suggested the following routine could be written in such a manner as to require less memory, or be written in a more efficient manner. The reader is invited to respond to Joe's suggestion, as well as encouraged to submit for publication other routines they have written which the novice programmer might find useful.

This program will assemble eight bit words from digits 0 through 7 on any ASCII encoded keyboard. It is also easy to add "Convenience Functions" like the error deletion subroutine I placed at address 076 (octal).

I am building up my system, piece-by-piece (like most of us) and wanted to do away with all the switch flipping as soon as possible. I purchased a 3 pxS I/O Port form Processor Technology, a couple of group purchases ago, that made the actual Altair-keyboard interfacing very straight forward. Because I bought the most inexpensive keyboard I could find, I had to add an eight bit data latch. I also added keyboard debouncing on the same PC board as the eight bit latch. If anyone is interested in adding these to your CHEAP keyboard give me a call. (I will include my phone number at the end.)

The program depends on the eighth bit for detecting a key-pressed condition. (The KP line out of the keyboard must be connected to the eighth bit position through your I/O Port—keypressed equals eighth bit low). The rest is not very difficult to understand from each of the notes in the program.

ASCII contains, at the far right of each eight bit code, the BCD (Binary Coded Decimal) equivalent of each number key. It is a simple matter to employ an AND instruction to strip away all unwanted bits leaving only the BCD portion at the right of the word. The hardest part is getting all three octal entries into one word. This requires that the first entry be shifted to the left 6 places, the second entry shifted 3 places to the left and the third entry remain at the far right. These are added into a previously cleared E register as they complete their shifting. After every third entry (complete word) the contents of the E register is loaded into memory, then the memory address register (H&L) pair is incremented by one.

I am not a programmer, merely a student of computer design with an Altair 8800. If anyone wants any additional info feel free to call me at 714/877-3006 or write me at 9866 Williams Ave., Bloomington, California 92316.

```
000          LXI H          041
001                         200
002                         000
```

Starting address of program to be written into memory with this loader. (Can be any area of memory you choose. Just put starting address here)

```
003          LXI B          001
004                         003
005                         006
```

Loads 006 into B register and 003 into the C register for control of the length of various loops. The 003 in the C register limits word length to three octal digits.

```
006          MVI E          036
007                         000
```

This clears the E register for use as an accumulator.

```
010           IN            333
011                         000
```

This inputs data from your keyboard (I have used 000 as my keyboard device No.)

```
012           CPI           376
013                         200
```

This checks to see if an entry is being made from the keyboard.

```
014           JNC           322
015                         010
016                         000
```

If a key was not being pressed at the IN instruction the program counter will jump back to 010 (IN). If a key has been pressed the program continues.

```
017           JMP           303
020                         076
021                         000
```

This is an optional branch for control functions that may be added to this program. I have added my error deletion program starting at 076.

```
022          ANI          346
023                       007
```

This strips all un-wanted prefixes from the ASCII leaving only the BCD (Binary Coded Decimal)

```
024          MOV AtoD     127
```

This stores the binary number from last keystroke in D.

```
025          MOV CtoA     171
026          CPI          376
027                       003
```

This retrieves the number previously loaded into C for comparison. (This number will be decremented by one each time the program loops through).

```
030          JZ           312
031                       042
032                       000
```

If this is the first loop, the number in C will be equal to the value compared. If so, the program will resume address at 042.

```
033          CPI          376
034                       002
035          JNZ          302
036                       072
037                       000
```

If this is the third loop through program (third key stroke), then the program will continue at address 072. If not the third key entry then it must be the second entry. This time the program continues to next address.

```
040          MVI B        006
041                       003
```

Since the value present in the B register is used to control the number of times each of the first two entries

```
042          MOV DtoA     172
043          RLC          007
044          DCR B        005
045          JNZ          302
046                       043
047                       000
```

will be shifted to the left, a three is needed for 2nd loop. Retrieves last key entry and shifts to the left until the B register is decremented back to zero.

```
050          ADD E        203
051          MOV AtoE     137
```

Add the contents of the E register to A. Store this new value in E register.

```
052          IN           333
053                       000
054          CPI          376
055                       200
056          JC           332
057                       052
060                       000
```

Check to see if the last key is still being pressed? If so, loop here until key is let up.

```
061          DCR C        015
```

Decrement the C register. (This keeps track of key entries, which must not go beyond THREE).

```
062          JNZ          302
063                       010
064                       000
065          MOV EtoM     163
066          INX H        043
```

If the C register decremented back to zero, then this was the last digit of three, now the complete word is moved into memory and the Memory Register is incremented to next address.

```
067          JMP          303
070                       003
071                       000
```

Jump back to start for the first digit of the next word.

```
072          MOV DtoA     172
073          JMP          303
074                       050
075                       000
```

The third entry doesn't need to be shifted to the left, since it is already in the least.significant position (the far right of the register) where it is needed.

Now the third entry can be added to the first two (which already reside in the E register) so the program counter is set for 050.

```
076          CPI          376
077                       030
100          JNZ          302
101                       022
102                       000
103          IN           333
104                       000
105          CPI          376
106                       200
107          JC           332
110                       103
111                       000
112          MOC CtoA     171
113          CPI          376
114                       003
115          JNZ          302
116                       003
117                       000
120          DCX H        053
121          JMP          303
122                       010
123                       000
```

This subroutine decrements the memory register back to the last address when the cancel key is pressed. If the cancel is desired in the middle of the entry of a word (1st or 2nd entry error) then the Address remains the same and the next writing will be the start of same address.

THE MAKINGS OF A MINI

Reprinted courtesy of DIGITAL DESIGN Magazine.

Not all computer hobbyists are technicians experienced in the world of solder, wire and PC boards. Many are sweatshirt and tennis shoe types (programmers, including this writer) who need an operator's manual to avoid burning themselves with solder.

Others will be newcomers to the field of electronics who simply want to purchase and learn to use a computer in the most economical manner possible—usually by buying a kit rather than the completed unit.

It's for the benefit of the last two categories of (potential) computer hobbyists that the following article is being reprinted from the August issue of DIGITAL DESIGN.

Pay particular attention to the sixth paragraph—regarding the broken wires. Your editor encountered the same difficulty and learned only afterward that the correct size of wire strippers and the use of harness lacing (even though only temporarily) can help minimize the problem discussed.

Altair 8800 Computer Kit. *MITS, Albuquerque, N.M. Price: $439.*

It's hard to be blasé about building your own computer. No matter how jaded you think you are, the arrival of that big box from the kit company brings out your most child-like impulses. First you want to tear open the carton and begin construction immediately. Like a kid on Christmas morning, you can't even take the time to use scissors or a knife to cut the wrappings. Instead, you shred the skin of your hands trying to rip away the tough fiber-glass tape. You're all set to skip dinner and begin work, but then you notice all those bags full of parts. Reason sets in. After all, a minicomputer is a complex piece of equipment, and a single, small mistake will blow the whole project. Better to approach the job logically and begin by reading the instruction manual.

The documentation MITS provides with each of its computer kits is extensive. The loose-leaf binder holds 83 pages of assembly and checkout procedures, a 92-page operator's manual and a 31-page section encompassing the unit's theory of operation and schematic diagrams. To keep assembly mistakes to a minimum, the book also includes hints on component handling, soldering and the use of tools.

Because the Altair 8800 is billed as a minicomputer cheap enough to use in the home, I felt that its construction should be a family affair. Though I had built stereo and amateur radio equipment from kits, my wife—with no prior electronic experience—proved much more proficient at inserting integrated circuits and other components into the small holes in printed circuit cards. After a little practice, she also became better than I at soldering leads to the cramped pads in the dense boards. But not wishing to be responsible for errors, she insisted that I translate the instructions and show her where each part went.

We began with the front-panel display/control board, the most complex and difficult to assemble. All went smoothly until we reached integrated circuit E, which we stupidly soldered in place with its polarity notch pointed in the wrong direction. In vain, we tried to unsolder it—a process that produced burned fingers and a mashed PC board. Ultimately, with diagonal cutters, we clipped each of the IC's leads and desoldered each pin individually. At a local electronics distributor, we bought a replacement, which we then soldered in properly. This six-hour object lesson in the perils of sloppiness served to underscore the basic rule of kit building: check, recheck and check again. Though the remaining electronics went smoothly, the mechanical wire connections proved troublesome.

Connecting the control panel to the rest of the computer requires that 63 loose wires be soldered to the board. Each of the wires, pre-cut in three-foot lengths, must be inserted into the proper holes in the PC board and then labeled with a piece of masking tape.

Though the mounting procedure is easy to follow, the stress imposed on the loose wires caused some of them to break off at their solder joints. Reinserting them took almost an hour. These same wires again proved troublesome when we had to insert them into the mini's motherboard. Each wire had to be cut to a different length, relabeled (because we had put the markets too close to the ends), and inserted into a board with a closely-spaced etch pattern. Once again, stress on the wires broke some of them off.

After the assembly of the front-panel, the CPU board, the memory board and the power supply went together easily.

Completing the assembly, we checked all the power points with a voltmeter. Every one registered properly. Then we gave the computer its first operating test. The wrong status lights came on. Trying every possibility we could think of, we could not make the *Address 5* light go out. So we began looking for wiring mistakes and solder bridges. To no avail.

After discussing the problem with an engineer at MITS, we shipped the computer back to them for repair. Within three weeks, they returned it in working order. According to the troubleshooter, there were three problems: a defective IC on the control board, a defective CPU chip and some bad soldering on my part,

(Continued on page 56)

The First 100 Days

A Report From The President

I just read the October, 1975 INTERFACE birth announcement of the Southern California Computer Society. It is rather sobering to look at John Walden's glossy words and to translate those high expectations into actions. This seems to be an appropriate time for a bench mark to see where we are and where we are headed.

As with all new-borns our first and most immediate problem was survival. We inherited very little from the Steering Committee in the way of on-going activities. To cope with the instantaneous situation we adopted short term expediencies to cover the balance of Calendar 1975, in order to gain time to plan for 1976. The plan of action for 1976 will cover the entire year thus permitting the succeeding administration a more orderly transition opportunity.

The practical aspects of organizing this administration quickly illuminated the divergent interests and philosophies of the Officers and Directors. The spirit of detente that prevailed throughout the lifetime of the Steering Committee scarcely survived ten minutes. During these ten minutes the Directors unanimously created the honorary offices of Chairman and Vice Chairman of the Board of Directors. I appointed Jerry Silver and Pearce Young, respectively, to these offices. Then came the deluge.

The basic philosophical questions dividing the Directorship center on the undefined goals of the Society. Some of these questions are:

1. Do we want the majesty and prestige of a formal organization, or the elegance of informality?
2. Should the membership door be wide open, or should we expand in a more orderly manner?
3. Will the Administration exercise its judgement and leadership, or will it seek advice and guidance from the membership?
4. Are we an association of experts and professionals in the computer field, or a collection of enthusiastic and irreverent amateurs?

In simple mathematical terms any set of yea's or nays to the above four questions has a 15-1 possibility of provoking opposition. I have insured myself of uniform criticism by pursuing aggressively the path of "all of the above". I have a priceless collection of communiques of this form:

Dear Sir,
You cur. A nasty letter will follow.
Sincerely,
(Name available on request)

These have invariably given me an insight into another perspective of the Society, and a renewed confidence we are not operating in a vacuum.

Independent of the adversities under which the Board has labored, their brilliance is reflected in their immense accomplishments. Your judgement will be measured against your hopes and expectations, and I trust your comments and criticisms will be clearly expressed. I had originally intended to list and summarize the works in progress, but if the results are not self evident, then such a statement would serve no purpose.

In conclusion let me make this perfectly clear: it is premature to make any conclusions. In the spirit of this multiple choice report, I will close with a multiple choice of quotations:

"We have only just begun . . ."

or

"We have not yet begun to fight."

Respectfully yours,
Ward Spaniol

TELEPRINTER MAINTENANCE/REPAIR PREVENTATIVE MAINTENANCE

PART 2

by Cliff Sparks

This is the second article in a series describing the operational functions of the 33 ASR/KSR.

Let's take a close look at the teleprinter character and its structure. The character is made up of eleven bits of information that are current and no current time intervals sent to the line. These intervals are referred to as Marking (current flow) and Spacing (NO current flow). Let's break down each of the eleven pulses and look into them. The first pulse is always Spacing and it is the START pulse. The next eight pulses are either Marking (current flow) or Spacing (NO current flow), let's refer to these special eight pulses as intelligence pulses. The last two pulses are the stop pulses and they are always marking.

FUNCTIONAL GROUPS. The Model 33 ASR contains five functional groups—five portions, each of which can be removed as a unit from the machine.

These functional groups are shown in a Block Diagram in view 1. You can see the relationship between the groups and the complete set. The five functional groups include: the Keyboard, Typing Unit, Tape Punch, Tape Reader, and Call Control Unit. These groups, though each is a separate unit, are operationally interdependent upon each other; and, in most cases, the machine cannot operate correctly without all of them being present.

In the following text we give you a general introduction to each of the five functional groups. This text is intended to give you a better understanding of the purpose of each group and the relationship of the groups to each other.

Keyboard Group. The keyboard group contains all of the mechanisms required to change the mechanical action of depressing a key into electrical impulses of the teletypewriter eight-unit code. These electrical impulses are transmitted through interconnecting cables to predetermined terminals on the distributor plate. The keyboard output is a parallel output to the distributor plate located in the Typing unit.

Keyboards used in the Model 33 ASR teletypewriters that are compatible with most computer terminals, generate the ASCII (American National Standard Code for Information Interchange) code. The code and correct keyboard combination allow the operator to select the desired parity. Parity, simply defined, as programming the keyboard in such a manner that ASCII characters are generated in either even or odd groups. A nonparity keyboard will have the eighth pulse always marking. A parity keyboard allows the eighth pulse to change so that an even number of marking pulses are transmitted for every

Photo courtesy of Teletype Corporation, Skokie, Ill.

character. Let's take a very close look at the teletypewriter keyboard itself. It has forty-nine key tops, one space bar, and possibly a HERE IS key top. To further break down the forty-nine keys: twenty-six are alpha keys, ten are numeric, four are functions (machine functions), fourteen keys are control functions, six are symbols. We further discover that by depressing the Shift Key we obtain 21 additional characters for a grand total of 81 characters.

When a selected key is depressed the downward movement is translated by a codebar mechanism into a mechanical set-up corresponding to the code combination selected. Eight pairs of codebars whose functions are to set up eight pairs of contacts in the selected code arrangement. Example: Let's select the "A" intelligence pulse and see what happens.

1. Press the "A" keytop.
2. The "A" codebars in the keyboard slide into predetermined arrangements.
3. The eight intelligence contact wires on the keyboard contact block align themselves in such a manner that the number one and number seven contacts are marking.
4. Cables connected to the contact block wires are routed to the distributor disc segments within the typing unit. When the distributor disc receives the marking pulse, in parallel form, it is routed on to segments cut on the disc face (see view 2).
5. We now have marking pulses on segments one and seven, awaiting transmission. The Teletypewriter is so designed that the next step in transmitting out "A" character requires only mechanical power. The power is available and being checked until it is required. A universal mechanism in the keyboard trips the distributor clutch in the typing unit, allowing the distributor brush holder to sweep the face of the disc in a 360 degree motion. This action causes the marking pulses on segments one and seven to be transmitted to: a) The line. b) The selector magnet. c) The selector mechanism in the typing unit.

Typing Unit Group. This group is made up of the motor, selector magnet, selector mechanism, distributor mechanism, Codebar mechanism, Function mechanism, Print carriage, Spacing mechanism, Carriage Return mechanism, Answer-Back mechanism.

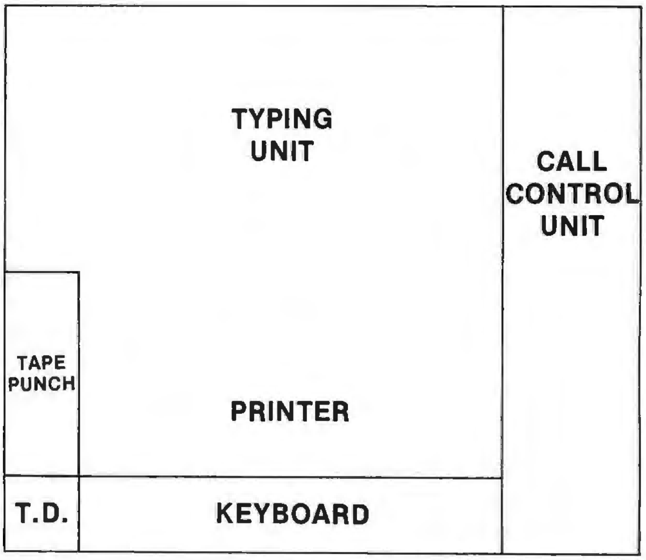
The motor group, which is the primary source of mechanical power, is mounted in the center and to the rear of the unit as viewed from the keyboard. The synchronous motor needs no adjustment and maintains an exact speed of 3600 RPM, with a power input of 60 Hz a.c.

The selector magnet group operates with a 500 milliamp loop, between the magnet and the selector magnet driver card located in the Call Control Unit.

The selector mechanism receives the intelligence coded information from the selector magnet driver and converts this information into mechanical pulses that control the codebar mechanism.

The distributor mechanism trips each time a keytop is depressed, allowing the segment disc (distributor) to sweep the segment face, looking for coded information.

The Codebar mechanism is tripped by the selector mechanism and the codebar clutch assembly. There are ten codebars in the unit and their function is to sense the marking and spacing position of the blocking lever in the selector mechanism. Function levers positioned under the codebars are put under spring tension in such a manner that they seek upward movement. Slots cut into the function levers seek special codebars during the operational cycle. The alignment of function lever and codebar determine the intelligence code being transmitted. CONFUSED, WORRY NOT FOR WE WILL EXPLAIN THESE MECHANISMS IN DETAIL AGAIN AND AGAIN.



VIEW 1.

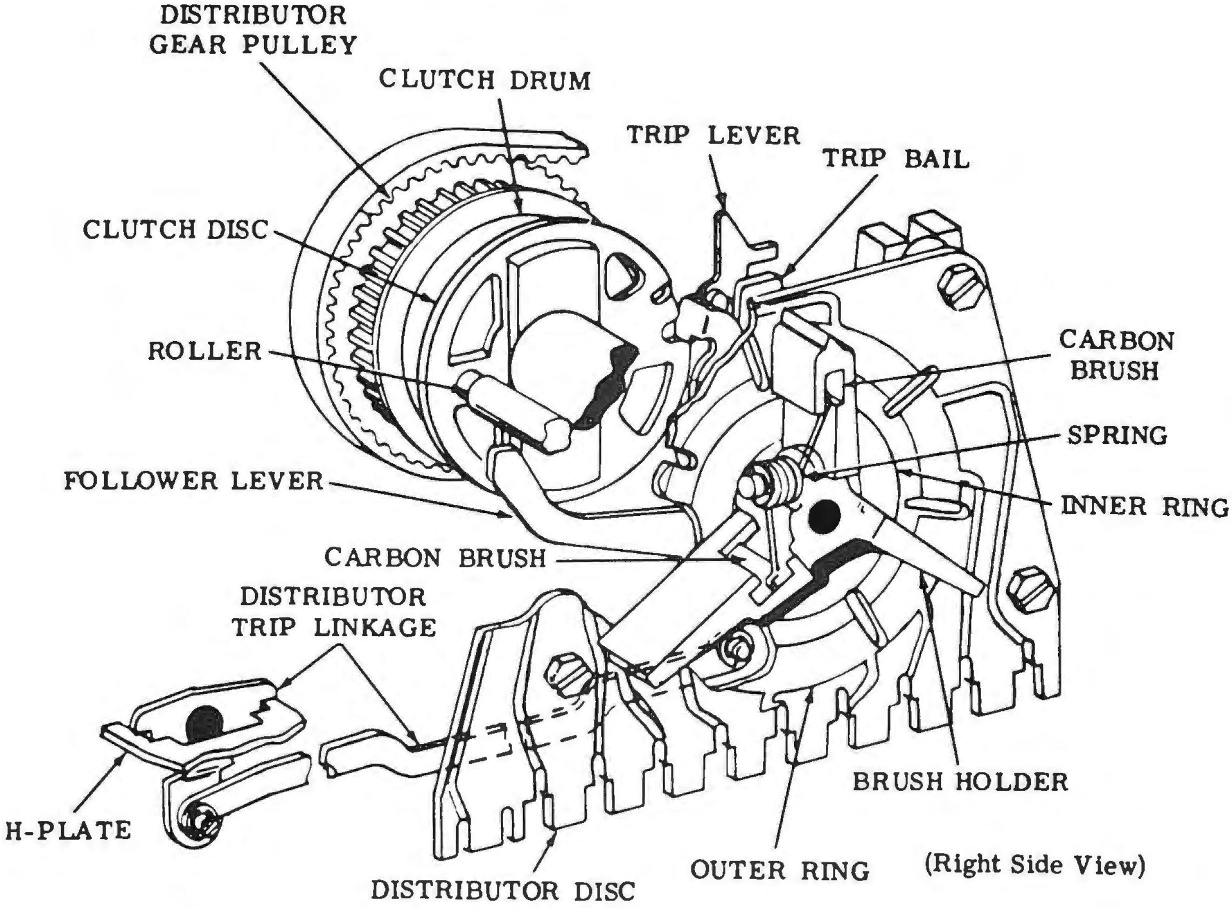
The print carriage mechanism is the mechanism which does the actual printing of a character. This group is mounted dead center of the printer and traverses to the right until the end of line area is reached, whereupon it is smartly returned to the left by the carriage return spring and carriage return mechanism. This group contains the type wheel, ribbon reverse mechanism, connecting bars (stop slides), print reset arm, trip lever, power bail and print hammer.

The spacing mechanism group acts as the device that spaces typed characters evenly.

The carriage return group returns the carriage to the left margin through the operation of the carriage return key top on the keyboard.

The answer-back mechanism is an optional device that may be programmed by the operator to transmit a predetermined sequence of characters for identification purposes.

The Tape Punch Group is strictly a mechanical input device and has no electrical connections. It is slaved from the Typing Unit and will reproduce on one inch tape, the exact character cycled through the Typing Unit. Extensions on the typing unit codebars



lock themselves into the tape punch extension codebars. The locking action causes the tape punch codebars to reproduce any character codes found on the typing unit. The punch is powered by a take-off arm mounted on the typing unit, as each coded character is cycled through the typing unit codebars the tape punch codebars set up the coded character and the tape punch drives the punch pins to perforate holes in paper tape.

The tape reader has eight sensing pins that are driven upward each cycle of operation. When the paper tape has been prepared (punched) it may be placed in the tape reader and transmitted. As the paper tape passes over the sensing pins in the reader, the pins pop through a punched hole (coded intelligence pulse). The sensing pins are connected to contact wires within the paper tape reader, as the tape passes over the sensing pins, holes are sensed, contact wires are made to make or break on a contact bar wired through a connecting harness to the distributor disc. The mark or spacing signals received from the tape transmitter (TD), are then transmitted to the line and the selector assembly.

The Call Control Unit is the switchboard of the teletypewriter and contains most of the wiring and electrical components of the machine.

The power input cord is connected through a fuse to the on and off switch. This power switch controls all power to the machine. Located on the right front of the printers cabinet it is a three position switch. Turning the switch full clockwise puts the printer in the local position and allows for private operation that will not interfere with on-line transmissions. Full counter clockwise puts the printer in the on-line mode. The center or twelve o'clock position turns the printer motor off. CAUTION: Power is still applied to the Call Control Unit and may only be removed by disconnecting the a.c. power cord from the wall outlet.

The Call Control Unit also contains the Selector Magnet Driver Card (SMD), Power Supply, Molex connectors (anywhere from a maximum of eight to six), line relay circuit, and fuses.

A disassembled teletypewriter will be displayed at the next SCCS meeting, at that time we will attempt to supplement this article by more detailed explanation of the 33 ASR. □

A Lesson in Economics

What happens when a good becomes so desirable and sought after that many people want to have it? Unless an ample supply is forthcoming, the price can and must go up (political fiat to the contrary notwithstanding). Recently we wondered why one distributor kept asking us for more and more of BYTE Number 1. He was very insistent, and spent several transcontinental telephone calls urging us (in vain) to send him more of BYTE's inaugural issue.

We found out recently, thanks to several callers, that this fellow was selling BYTE's Number 1 issue for $15.00 each, since everybody wants one and the supply is very limited. Of course he made a temporary windfall profit on his early investment in a supply of the new magazine. He paid what everyone else did who purchased a bulk shipment of the then unknown magazine. There is a law of economics that price rises when supplies are tight – it is as much a folly to fight such a trend as it is to attempt to legislate Pi = 3.0000... Many thanks to Adam Smith and Ludwig von Mises for tipping us off about the law of supply and demand, confirmed again by this laboratory experiment.

The point of mentioning this is to put to rest rumors about our supposed reprinting of the early issues of BYTE. First, we are not planning to reprint BYTE magazine issues as magazines, ever. You'll see individual articles reappearing in books of selected reprints which we plan to publish. However our early supporters will see their confidence rewarded as the price of the first issue collector's item rises over the years, since we will never reprint it as a magazine. Our policy is now and will continue to be one of matching our press runs to the subscription and newstand demand, with very few extra issues kept available. If you want to keep up with this fast moving field, you'll have to keep your subscription current.

Cut

Name ....

Address ....

City .... State .... Zip ....

□ BILL ME □ Check for $12 enclosed
□ Bill BankAmericard or MasterCharge # ....

Tear

Name ....

Address ....

City .... State .... Zip ....

□ BILL ME □ Check for $12 enclosed
□ Bill BankAmericard or MasterCharge # ....

Shred

Name ....

Address ....

City .... State .... Zip ....

□ BILL ME □ Check for $12 enclosed
□ Bill BankAmericard or MasterCharge # ....

Rip

Name ....

Address ....

City .... State .... Zip ....

□ BILL ME □ Check for $12 enclosed
□ Bill BankAmericard or MasterCharge # ....

• BYTE • 70 Main St • Peterborough NH 03458

4K Memory Check

EDITOR'S NOTE: A close look at Jon's Processor Technology Memory Board test program will reveal the reversal of the register designations in many instructions. Jon commented, as he gave us the program, that he had been reading left to right since childhood and could see no point in changing now.

It does seem awkward to say "move A to B" while writing "MOV B, A." Most 8080 programmers have been thinking (if not saying—mumbling?) "move to B (pause) A" long enough to simply shrug their shoulders and continue coding. Jon on the other hand, is relatively new to assembly language programming, and not biased by hardware engineering training. His simple solution to a problem perceived by an unspoiled mind might be worth thinking about the next time an assembler is written.

P.T. 4K MEMORY CHECK

DESCRIPTION: This program is a two-part memory diagnostic written to test Processor Technology's 4K memory boards (one at a time) and is applicable to any 4K board using 1024 x 1 bit RAM chips.

Part I tests the ability of memory to accept zeros and ones. The configuration being stored is held in the E register and is referred to as "check". Zeros are stored first and compared. They are next changed to ones and compared, and finally the ones are changed back to zeros and compared. The three runs are kept track of in register D. When the "1" is finally rotated into the carry bit, all three runs are completed.

Part II has two phases: Phase one walks a "1" bit through a field of zeros; phase two walks a "0" bit through a field of ones. The "phase" is kept in register D; its address (the address for comparison) is in H & L. The walking "bit" is kept in register E; its address is in B & C. The process begins by storing 01 in address zero and checking all other addresses to see if they remained zero (unchanged). Address 00 is then returned to "phase" (in this case, zero) and the "bit" is moved to address 0001. The process continues until the "bit" has been stored in OFFF (the last word in 4K). It is then rotated (to become 02), and the cycle is repeated. After the eighth cycle (80), all memory words are changed to FF and the last phase begins with FE as the "bit". When the 7F cycle is over, the program is over.

ERRORS: When an error occurs, the needed data is pushed onto a stack and the program loops to itself until stopped (see OPERATION). After reading the stack data, the program can be asked to continue the test in progress, but at the end of that segment it will abort. For example, if an error occurs in Part I, you may continue to the end of Part I. Part II, however, will be skipped since its accuracy depends on knowing that both zeros and ones will be accepted by all of memory. If an error occurs in Part II, you may continue through the end of that "bit" cycle only. The rest of Part II will be skipped since returning the error to "phase" could itself create new errors.

In either case, the bad chips should be changed, and the program should be run again from the top. Errors can be rélated to chips by using P.T. schematic 4KRA (or the schematic for your board) or (for P.T.) by using the following chart:

	bits in error:							
address:	7	6	5	4	3	2	1	0
oooo - 01FF	8	7	6	5	4	3	2	1
0200 - 03FF	16	15	14	13	12	11	10	9
0400 - 07FF	24	23	22	21	20	19	18	17
0800 - 0FFF	32	31	30	29	28	27	26	25

Examples: If bit 4 is in error at address 05D3, replace IC21; if bit 2 is in error at 00FC, replace IC3; etc.

BOARD ADDRESS: The board to be tested should be addressed for the first 4K of memory (000 - OFFF). The program begins at 1000 and operates only on the first 4096 words as a unit.

PROGRAM MODIFICATION: Labels are used in the program to facilitate analysis and modification. If the program is placed somewhere else in memory, change:

1. All label addresses for jumps;
2. Operands for LXI SP (101A, 1072);
3. Operands for LDA (103E, 1042, 1080, 1084).

If the program is modified to begin at an address other than 0000, change operands for LXI B (100D, 100E, 104B, 104C, 10AC, 10AD) and LXI H (1051, 1052). If the program is modified for other than 4K, change operands for CPI (102C, 1067, 1093, 10B9).

RUN TIME: With no errors, run time is approximately 2.5 hours for 4K. (Again, this is for a P.T. board; other boards may vary from this.) Be sure to allow for the change in run time if the area tested is either increased or decreased. For nK it takes about 2^n x 9.5 minutes. I tested 2K in 38 minutes, 8 seconds. Checking 64K at once would take about 27 days!

OPERATION:

1. Enter program.
2. Examine 1008 (1ST) and run.
3. If loop is at 1021 (ERI),
   A. Read stack for error at 1004:
      1004 "check" (what the word should be)
      1005 "error" (what the word is)
      1006 low order error address
      1007 higher order error address
   B. Examine 1024 (OK1) and continue.
4. If loop is at 107B (ER2),
   A. Read stack for error at 1000:
      1000 "phase" (what the word should be)
      1001 "error" (what the word is)
      1002 low order error address
      1003 high order error address
      1004 low order bit address
      1005 high order bit address
      1006 "bit" (the word causing error)
      1007 "phase" (same as 1000)
   B. Examine 1053 (OK2) and continue.
5. If loop is at 10C5 (END), program is over. (See notes on ERRORS.)

```
ADDR  LABEL   1   2   3   OP CODE  OPERND  COMMENTS
1000          00  00  00                   }
1003          00  00  00                   } Stack
1006          00  00                       }
                                           PART I
1008  1ST     16  04      MVI D    04         Complement register (set up for 3 checks)
100A          1E  00      MVI E    00         Check register (start with zeros)
100C  NXT1B   01  00  00  LXI B    0000       Initial address at zero
100F  NXT1A   7B          MOV E-A             Set up check for storing
1010          02          STAX B              Store check
1011          60          MOV B-H          }  Save old address
1012          69          MOV C-L          }
1013          03          INX B               New address
1014          BE          CMP M               Did the check take?
1015          CA  24  10  JZ       OK1        Jump if it did
                                           ERROR OCCURED
1018          31  08  10  LXI SP   1ST        Set stack pointer
101B          E5          PUSH H              Error address
101C          66          MOV M-H             Error ready to push
101D          6F          MOV A-L             Check ready to push
101E          E5          PUSH H              Error & check
101F          E1          POP H            }Retrieve address for re-entry
1020          E1          POP H            }
1021  ER1     C3  21  10  JMP      ER1         Loop till stopped
```

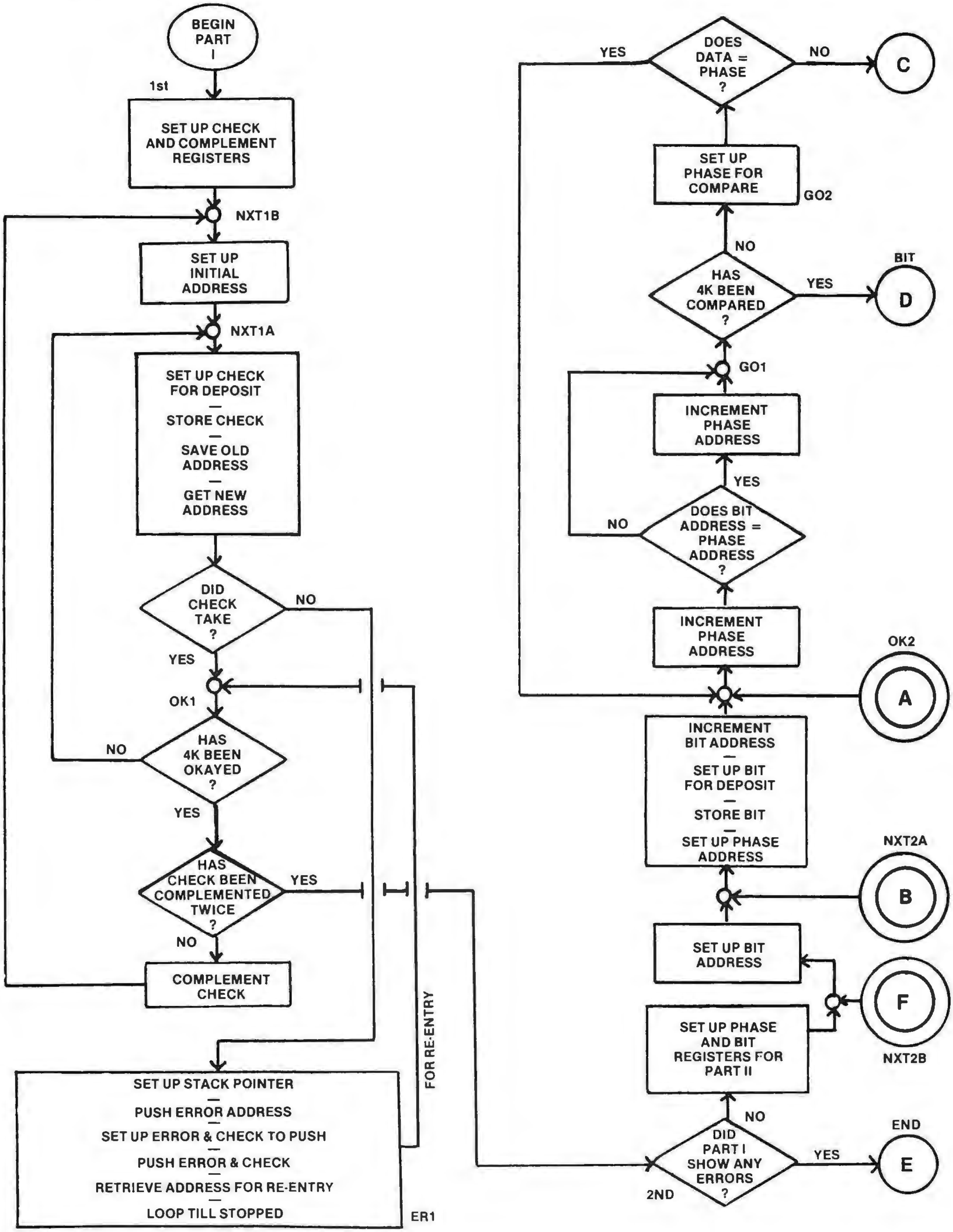
```
ADDR  LABEL   1   2   3   OP CODE   OPERND      COMMENTS
                                                NO ERROR OCCURED
1024  OK1     79          MOV C-A                Compare new address with 1000 to see
1025          FE  00      CPI       00           if all 4K has been checked; if not,
1027          C2  0F  10  JNZ       NXT1A        loop back for next phase check
102A          78          MOV B-A
102B          FE  10      CPI       10
102D          C2  0F  10  JNZ       NXT1A
                                                4K HAS BEEN CHECKED
1030          7A          MOV D-A                Has check been complimented twice?  (1
1031          1F          RAR                    will rotate into carry if it has)
1032          DA  3C  10  JC        2ND          Part I is completed
1035          57          MOV A-D                Part I isn't completed; put new number in
1036          7B          MOV E-A                complement register; call in check and
1037          2F          CMA                    complement it; put new check in check
1038          5F          MOV A-E                register
1039          C3  0C  10  JMP       NXT1B        Loop back for new check
                                                SHOULD PART II BE DONE?
103C  2ND     3A  05  10  LDA       3rd stack    Call in error and check words from stack;
103F          67          MOV A-H     word       if they aren't equal, there has been a
1040          3A  04  10  LDA       4th stack    problem in storing bits; don't bother
1043          BC          CMP H       word       with Part II until changing bad chips
1044          C2  C5  10  JNZ       END
                                                PART II
1047          56          MOV M-D                Clear phase register
1048          1E  01      MVI E     01           Bit register (start with 01)
104A  NXT2B   01  FF  FF  LXI B     FFFF         Initial address at zero for storing
104D  NXT2A   03          INX B                     (called bit address)
104E          7B          MOV E-A                Set up bit for storing
104F          02          STAX B                 Store bit
1050          21  FF  FF  LXI H     FFFF         Initial address at zero for comparing
1053  OK2     23          INX H                     (called phase address)
                                                AVOID COMPARING TEST WORD
1054          79          MOV C-A                Compare bit address with phase address;
1055          BD          CMP L                  if they are different, jump and go on
1056          C2  5F  10  JNZ       GO1
1059          78          MOV B-A
105A          BC          CMP H
105B          C2  5F  10  JNZ       GO1
105E          23          INX H                  If they're equal, skip comparing
                                                HAS 4K BEEN COMPARED?
105F  GO1     7D          MOV L-A                Compare phase address with 1000 to see
1060          FE  00      CPI       00           if all 4K has been compared; if it has,
1062          C2  6B  10  JNZ       GO2          find out what the bit address is
1065          7C          MOV H-A
1066          FE  10      CPI       10
1068          CA  7E  10  JZ        BIT
                                                4K NOT COMPARED YET
106B  GO2     7A          MOV D-A                Set up phase for compare
106C          BE          CMP M                  Did the word remain unchanged?
106D          CA  53  10  JZ        OK2          Loop back for next if it's OK
                                                ERROR OCCURED
1070          31  08  10  LXI SP    1ST          Set stack pointer
1073          D5          PUSH D                 Phase & bit
1074          C5          PUSH B                 Bit address
1075          E5          PUSH H                 Phase address
1076          66          MOV M-H                Error ready to push
1077          6A          MOV D-L                Phase ready to push
```

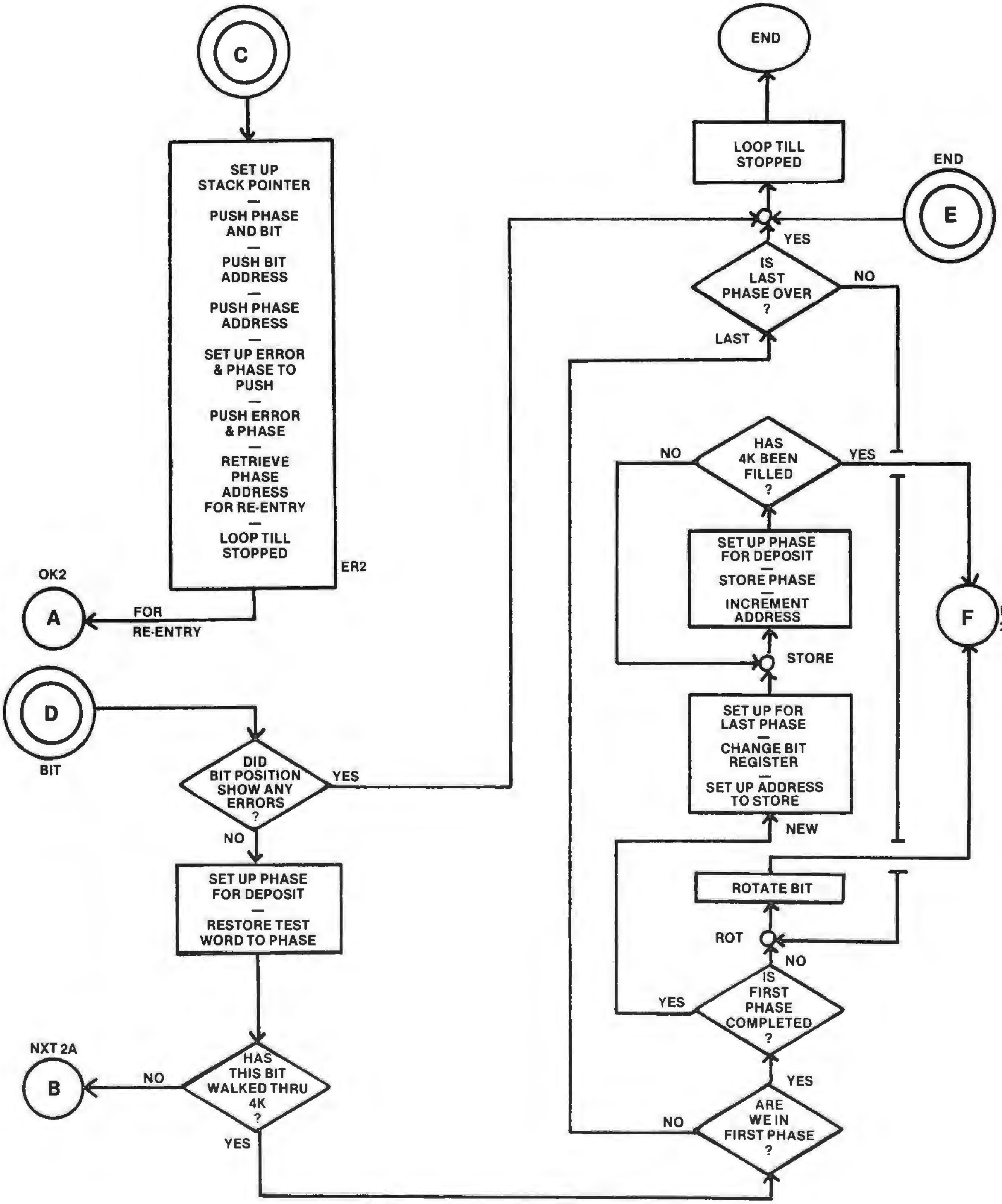
```
ADDR  LABEL   1   2   3   OP CODE   OPERND       COMMENTS
1078          E5          PUSH H                  Error & phase
1079          E1          POP H                   Retrieve address for re-entry
107A          E1          POP H
107B  ER2     C3  7B  10  JMP       ER2           Loop till stopped
                                             -SHOULD PART II BE CONTINUED?
107E  BIT     3A  01  10  LDA       7th stack     Call in error & phase words from stack;
1081          67          MOV A-H     word        if they aren't equal, there has been a
1082          3A  00  10  LDA       8th stack     problem in generating extra bits or in
1085          BC          CMP H       word        dropping them; don't bother checking
1086          C2  C5  10  JNZ       END           further until changing bad chips
                                             -CONTINUE PART II
1089          7A          MOV D-A                 Set up phase for deposit
108A          02          STAX B                  Restore the word to phase
                                             -HAS BIT WALKED THROUGH 4K?
108B          79          MOV C-A                 Compare bit address with 0FFF to see
108C          FE  FF      CPI       FF            if bit has walked through all 4K; if
108E          C2  4D  10  JNZ       NXT2A         it hasn't, loop back for next test
1091          78          MOV B-A                 word
1092          FE  0F      CPI       0F
1094          C2  4D  10  JNZ       NXT2A
                                             -WHAT PHASE ARE WE IN?
1097          AF          XRA                     Clear A
1098          BA          CMP D                   Are we in first phase?
1099          7B          MOV E-A                 Set up for later bit compare
109A          C2  C0  10  JNZ       LAST          Jump if we're in last phase
                                             -FIRST PHASE
109D          FE  80      CPI       80            Has first phase been completed?
109F          CA  A7  10  JZ        NEW           If so, set up for last phase
                                             -PHASE INCOMPLETE
10A2  ROT     07          RLC                     Rotate into new bit position
10A3          5F          MOV A-E                 Put new bit position in bit register
10A4          C3  4A  10  JMP       NXT2B         Loop back for new bit walk
                                             -SET UP FOR LAST PHASE
10A7  NEW     16  FF      MVI D     FF            All ones in phase register
10A9          1E  FE      MVI E     FE            Bit register (start with FE)
10AB          01  00  00  LXI B     0000          Initial address at zero (for storing)
10AE  STORE   7A          MOV D-A                 Set up phase for storing
10AF          02          STAX B                  Store phase
10B0          03          INX B                   New address
                                             -IS 4K FILLED WITH LAST PHASE?
10B1          79          MOV C-A                 Compare address with 1000 to see if
10B2          FE  00      CPI       00            4K has been filled with phase; if
10B4          C2  AE  10  JNZ       STORE         not, loop back to continue storing
10B7          78          MOV B-A
10B8          FE  10      CPI       10
10BA          C2  AE  10  JNZ       STORE
10BD          C3  4A  10  JMP       NXT2B         If so, loop back for new bit walk
                                             -LAST PHASE
10C0  LAST    FE  7F      CPI       7F            Has last phase been completed?
10C2          C2  A2  10  JNZ       ROT           If not, jump to get new bit position
10C5  END     C3  C5  10  JMP       END           If so, loop till stopped

10C8  (begin free memory)
```

P.T. 4K MEMORY CHECK

(1)

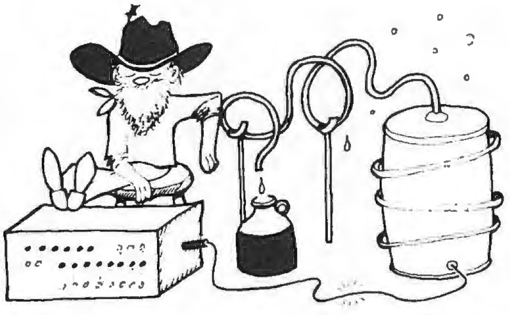

BEGIN PART I
1st
SET UP CHECK AND COMPLEMENT REGISTERS
NXT1B
SET UP INITIAL ADDRESS
NXT1A
SET UP CHECK FOR DEPOSIT — STORE CHECK — SAVE OLD ADDRESS — GET NEW ADDRESS
DID CHECK TAKE ?
NO
YES
OK1
HAS 4K BEEN OKAYED ?
NO
YES
HAS CHECK BEEN COMPLEMENTED TWICE ?
YES
NO
COMPLEMENT CHECK
FOR RE-ENTRY
SET UP STACK POINTER — PUSH ERROR ADDRESS — SET UP ERROR & CHECK TO PUSH — PUSH ERROR & CHECK — RETRIEVE ADDRESS FOR RE-ENTRY — LOOP TILL STOPPED
ER1
DOES DATA = PHASE ?
YES
NO
C
SET UP PHASE FOR COMPARE
GO2
NO
HAS 4K BEEN COMPARED ?
YES
BIT
D
GO1
INCREMENT PHASE ADDRESS
YES
DOES BIT ADDRESS = PHASE ADDRESS ?
NO
INCREMENT PHASE ADDRESS
OK2
A
INCREMENT BIT ADDRESS — SET UP BIT FOR DEPOSIT — STORE BIT — SET UP PHASE ADDRESS
NXT2A
B
SET UP BIT ADDRESS
F
SET UP PHASE AND BIT REGISTERS FOR PART II
NXT2B
NO
DID PART I SHOW ANY ERRORS ?
2ND
YES
END
E


C
SET UP STACK POINTER
PUSH PHASE AND BIT
PUSH BIT ADDRESS
PUSH PHASE ADDRESS
SET UP ERROR & PHASE TO PUSH
PUSH ERROR & PHASE
RETRIEVE PHASE ADDRESS FOR RE-ENTRY
LOOP TILL STOPPED
ER2
OK2
A
FOR RE-ENTRY
D
BIT
DID BIT POSITION SHOW ANY ERRORS ?
YES
NO
SET UP PHASE FOR DEPOSIT
RESTORE TEST WORD TO PHASE
NXT 2A
B
NO
HAS THIS BIT WALKED THRU 4K ?
YES
END
LOOP TILL STOPPED
END
E
YES
IS LAST PHASE OVER ?
NO
LAST
HAS 4K BEEN FILLED ?
NO
YES
SET UP PHASE FOR DEPOSIT
STORE PHASE
INCREMENT ADDRESS
STORE
F
NXT 2B
SET UP FOR LAST PHASE
CHANGE BIT REGISTER
SET UP ADDRESS TO STORE
NEW
ROTATE BIT
ROT
NO
YES
IS FIRST PHASE COMPLETED ?
YES
NO
ARE WE IN FIRST PHASE ?

MITS Altair™ Computers
SWTPC TV Typewriter II
CROMEMCO TV Dazzler™

THE COMPUTER STORE 11656 W. Pico Blvd., Los Angeles, Ca. 90064
(213) 478-3168 Bankamericard and MasterCharge welcome

GAMES & THINGS



by Mike Stern

EDITOR'S NOTE: Mike asks that his address be printed to encourage all you computer game enthusiasts to submit your favorite for review. Please mail your contributions to:

Mike Stern, Games Editor
P.O. Box 44236
Panorama City, CA 91412

FAR OUT THINGS . . .

If you are really interested in games, I can recommend not one, but TWO! books for you to buy, borrow, or beg.

NUMBER ONE!!!

What To Do After You Hit Return or P.C.C.'s First Book of Computer Games. (My old, decrepit, High school English teach always said to underline titles, and boy! with a title like that, it seemed like forever!)

This book is out-a-sight! Just the cartoons are worth the ($6.95) price! It covers everything from number-guessing games to Star Trek. Buy it, you'll like it!

NUMBER TWO!!!

101 Computer Games.

This book is Digital Equipment's very own game book. Although there is some duplication between this book and P.C.C.'s, there are many, many other games. The important thing about this book, however, is that it has a COMPLETE listing for Star Trek! (P.C.C.'s doesn't, but it is more entertaining.)

NOW, A MAGAZINE RECOMMENDATION . . .

If you don't already have one, get a subscription to *Scientific American*. The "Mathematical Games" column often has ideas that are translatable into computer games.

DISPLAYS, DISPLAYS AND MORE DISPLAYS.

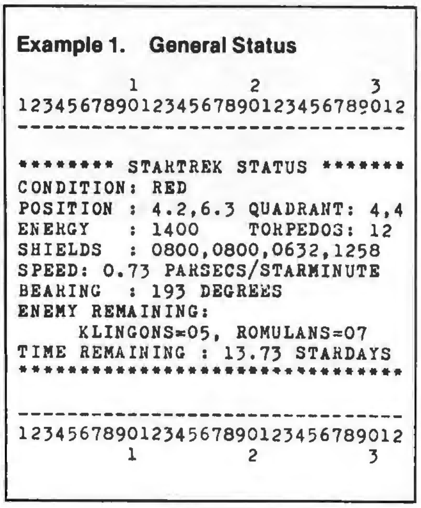
Let's talk about the displays that we need for Startrek. (After all, we can't write the program without knowing what the output looks like, can we?)

The displays can be broken down into two distinct categories: I) Status Displays and II) Location (mapping) displays. I have taken the liberty of changing the formats of the displays from 80 characters by 24 lines to 32 characters by 16 lines. The reason I did this is so that they will fit into a standard TVT display. (Coincidentally, this column is 32 characters across, but I swear that that had nothing to do with my choice of size.)

STATUS DISPLAYS

A) THE MAIN STATUS DISPLAY

The main display shows normal, ordinary, run-of-the-mill information that is necessary for playing a normal game. Information can include (but is most certainly not limited to) the following (see Example 1.):



Example 1. General Status

```
         1         2         3
12345678901234567890123456789012
--------------------------------

******** STARTREK STATUS *******
CONDITION: RED
POSITION : 4.2,6.3 QUADRANT: 4,4
ENERGY   : 1400    TORPEDOS: 12
SHIELDS  : 0800,0800,0632,1258
SPEED: 0.73 PARSECS/STARMINUTE
BEARING  : 193 DEGREES
ENEMY REMAINING:
     KLINGONS=05, ROMULANS=07
TIME REMAINING : 13.73 STARDAYS
********************************


--------------------------------
12345678901234567890123456789012
         1         2         3
```

1) General Status (CONDITION)

a) GREEN—there are no enemies around.

b) YELLOW—low on energy.

c) ORANGE—somewhere in your vicinity, an evil and nefarious Klingon or Romulan is lurking.

d) RED—someone has let loose a torpedo (are you in its path?).

2) Ship position.

The X and Y coordinates that the Enterprise is currently occupying in the quadrant. (You may want to include the X and Y coordinates of the quadrant itself.)

3) The current amount of energy.

4) The number of photon torpedos currently available.

5) The amount of power in your deflector shields.

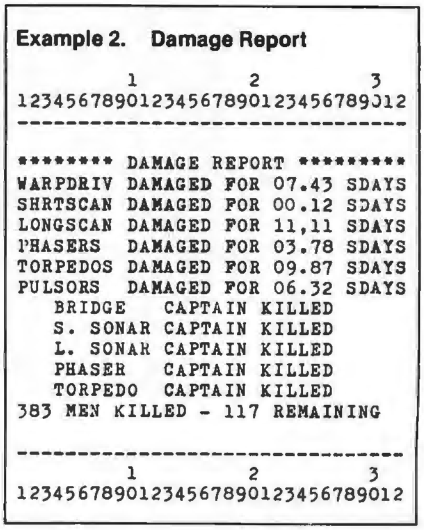
6) Your current speed and bearing.

7) The number of enemies left.

8) The amount of time remaining in the game.

B) DAMAGE REPORT

Specify the facility that has been damaged, and how long it will remain in its inoperable condition. If you cannot use a facility due to an individual's demise, be certain to include the information in your damage report also. A last (but very important) item that must be included is the number of men remaining aboard the Enterprise. (see Example 2.)



Example 2. Damage Report

```
         1         2         3
12345678901234567890123456789012
--------------------------------

******** DAMAGE REPORT *********
WARPDRIV DAMAGED FOR 07.43 SDAYS
SHRTSCAN DAMAGED FOR 00.12 SDAYS
LONGSCAN DAMAGED FOR 11,11 SDAYS
PHASERS  DAMAGED FOR 03.78 SDAYS
TORPEDOS DAMAGED FOR 09.87 SDAYS
PULSORS  DAMAGED FOR 06.32 SDAYS
   BRIDGE   CAPTAIN KILLED
   S. SONAR CAPTAIN KILLED
   L. SONAR CAPTAIN KILLED
   PHASER   CAPTAIN KILLED
   TORPEDO  CAPTAIN KILLED
383 MEN KILLED - 117 REMAINING

--------------------------------
         1         2         3
12345678901234567890123456789012
```

C) SHORT RANGE TRACK.

This is a useful display which shows the direction and speed of all dangerous objects in your quadrant. Super Startrek doesn't track Romulans to make them harder to do away with. (see Example 3.)

Example 3. Short Range Track

```
           1         2         3
12345678901234567890123456789012
--------------------------------

******* SHORT RANGE TRACK ******
OBJECT      SPEED  BEAR   COORDS
ENTERPRISE 0.73    193    4.2,6.3
KLINGON     0.43   110    3.3,8.9
TORPEDO     0.54    45    7.1,1.0
TORPEDO     0.02   360    1.9,9.0
TORPEDO     0.58   262    1.9,3.9

--------------------------------
12345678901234567890123456789012
           1         2         3
```

MAPPING DISPLAYS:

A) QUADRANT DISPLAY (short range scan).

This is a display of the current quadrant in which you are traveling (see Example 4.). Note that much of the information from the main status display is also on this screen.

Example 4. Quadrant Display

```
           1         2         3
12345678901234567890123456789012
--------------------------------

********* QUADRANT MAP *********
COND : RED  SPEED : 0.73 AT 193
TORPS: 12   ENERGY: 1200
SHIELDS: 0800,0800,0632,1258
  .  .  .  .  .  .  T  .  .  .
  .  .  .  .  .  .  .  .  .  .
  .  R  .  .  .  .  .  .  B  .
  .  T  .  .  .  .  .  .  .  .
  .  .  .  .  .  .  .  *  .  .
  .  .  .  E  .  *  .  .  .  .
  .  .  .  .  .  .  .  .  .  .
  .  .  .  .  .  .  .  .  .  .
  .  T  K  .  .  .  .  .  .  .
  .  .  .  .  .  .  .  .  .  .

--------------------------------
12345678901234567890123456789012
           1         2         3
```

Symbols used in the quadrant display:

'.' - an unoccupied parsec.
'*' - a star.
'B' - a base.
'E' - the Enterprise.
'G' - a ghostship.
'K' - a Klingon.
'R' - a Romulan.
'T' - a Torpedo.

B) LONG RANGE SCAN

This display shows your current quadrant and the ones immediately surrounding it in a conventionalized format RKBS: where R is the number of Romulans, K is the number of Klingons, B is the number of bases, and S is the number of stars. An E is placed to the right of the quadrant currently occupied by the Enterprise.

i.e. 1112E means that there is one Romulan, one Klingon, one base, and two stars in the quadrant currently occupied by the Enterprise (see example 5)

Example 5. Long Range Scan

```
           1         2         3
12345678901234567890123456789012
--------------------------------

******** LONG RANGE SCAN *******
QUADRANT 4,4


      2003  0010  0000
      0002  1112E 0103
      0003  0204  0004


********************************

--------------------------------
           1         2         3
12345678901234567890123456789012
```

C) GALACTIC SCAN

Shows the status of the galaxy that the Enterprise may move in. The rules for the display are the same as for the long range scan (above), except that **** means that a quadrant has not yet been scanned or visited. (see example 6.)

Example 6. Galactic Map

```
           1         2         3
12345678901234567890123456789012
--------------------------------

******** GALACTIC MAP **********

  **** **** **** **** **** ****
  **** **** **** **** **** ****
  **** **** 2003 0010 0000 ****
  **** **** 0002 1112E0103 ****
  **** **** 0003 0204 0004 ****
  **** **** **** **** **** ****
  **** **** **** **** **** ****
  **** **** **** **** **** ****
  **** **** **** **** **** ****
  **** **** **** **** **** ****


--------------------------------
           1         2         3
12345678901234567890123456789012
```

MESSAGES . . .

Try to make your messages humorous (if possible). For example (I.E. to you people that understand Latin), in my own Startrek, if I try to shoot a torpedo at myself, the program refuses and says

"YOU ARE SHOOTING AT YOURSELF, HOW CAN I TRANSFER OUT OF THIS CHICKEN OUTFIT?????"

NEXT COLUMN . . .

For my next column, I will not be talking about Star Trek (I thought I heard a sigh of relief?!?), but I will have some comments on a computer game that you can play WITHOUT! a computer (hee, hee).

Until then . . .

HAPPY TREKING (from that li'l old gamemaster, we)!

Bits + Bytes

by Mike Teener

By the time all of you read this, I should finally have my very own Altair 680. After three years of reading, writing, and talking about micros I finally took the plunge and ordered a computer for my personal use. It all seems very silly to some of my friends at work. I have free access to big, fast computers with lots of sexy features and software and a minicomputer with all kinds of grahics capabilities . . . why am I *buying* what is admittedly crude, limited and unsophisticated?

Why, indeed?

Well, because I like to get my hands dirty. And because for me, it really is "the world's greatest toy". I was always delighted that the world liked to pay me to play with computers; now that I'm finding other fun games to play (like management and writing) that are taking up more and more time, I'm willing to pay a little to stay in the first game. Besides, I always wanted to be able to open my desk drawer and pull out a computer, complete with flashing lights.

God knows why you people are so crazy to buy a computer or want to *play* with those obstinate creations. But I don't know why, and it's something I'm interested in, so do me a big favor and write me a note . . . if it's interesting I'll stick it in this column.

I just finished a big survey of all the microcomputer board systems now available. What a pain. There are over fifty companies in the market and more starting up every week. My advice to you is to stick with the established hobbyist companies for the next six months until the smoke clears. Most of the industry-oriented systems are pretty grossly overpriced by our standards. And frankly, there's no good reason for it. The best bang-per-buck ratio I found was the IMSAI stuff, which they intend to market both to the industry and to you and me. MITS and Sphere also advertise in trade journals, and their prices are lower than industry standards, particularly when you look at customer support. Dealing with a lot of the non-hobbyist companies is an exercise in frustration; no one seems to be able to answer the questions I ask in a reasonably straight fashion. They also do not have either the facilities or the will to help one very small customer with his/her little problems.

I hope you enjoyed the last issue of INTERFACE. Art Childs and the rest of us spent a memorable evening trying to find a small apartment in Hollywood, then trying to proof the incredibly bad copies of the galleys which turned out to be incomplete, anyway. Then Art got to shepheard it the rest of the way by himself. Bravo, Art.

MORE INDUSTRY NEWS:

The Monolithic Memories, Inc. people make memory chips and four-bit slice processors (sort of like a 74181 ALU with registers and data path control logic). They have taken four of these slice processors and microprogrammed them to emulate the Data General NOVA, a popular (6-bit minicomputer). This isn't the first time this kind of thing has been done; Keronix and Digital Computer Controls both market NOVA look-alikes. What's new is the price: the 600ns CPU board *kit* costs $475, 4Kx16 memory with memory timing and control is $500, and an additional 16Kx16 is $1400. Serial interface is $250. This isn't quite down to MITS' prices, but is damn attractive considering the CPU does most operations in 1.8 to 2.4 microseconds and has an extraordinary amount of software available for it.

RCA is having a sale to celebrate the bicentennial (so who isn't?) How about a 1K static SOS/CMOS RAM (*150*ns and *2*mw) for $17.76. Figuring on normal price erosion, you should be able to get those kind of specs for $5 next year sometime, and $3 in a couple of years.

There are three new memory devices coming out this year that you might look for. A 16K bit dynamic RAM. A 4K bit static RAM (no more refresh!) And a 32K bit ROM. The impact in the hobbyist market won't be felt this year except to drive down even further the price of "conventional" memories (1K statics and 4K dynamics). Still, it sure would be nice to have 16K bytes on one memory board using only 8 chips or a complete 8K basic on 2Ks.

Hey, out there in hobbyist territory, the semiconductor and microcomputer industry is taking you seriously. The latest series of Intel microcomputer boards is being marketed as being "ideal for the hobbyist". Microcomputer Associates' JOLT system was advertised in BYTE as many of you know —this was originally intended for evaluation and OEM use.

IMS Associates, a company marketing Altair 8800 compatible systems (CPU's, memories, I/O, even control panels and chassis) is getting a lot of press coverage over their new computer system, the Hypercube II. A totally outrageous, yet feasible design, the Hypercube is a network of independent nodes, each of which has two 8080's (one for communications and memory control and one for user computa-

(Continued on page 56)

HARDWARE REPORT

by S. Wilcox

Polymorphic Systems of Goleta, California is offering a new entry into the ALTAIR 8800 compatible interface field to fill a gap that was, to this time, left open. This board is the type that is considered the ultimate interface by some, an analog to digital and digital to analog converter. That's right an A/D and D/A on one board!

This board can interface your ALTAIR to the real world directly. Connect any devices that can produce a voltage change with the parameter being measured: photocells, thermisters, Hall devices, any analog transducer. Then connect X Y plotters, servos, speed controls, any analog device that can use a computer controlled output voltage.

The board has up to 2 analog outputs with 10 bits of resolution each, 6 digital control bits, and 8 analog inputs with combinations of these selectable by onboard jumpers. There is also a bipolar option available.

SPECIFICATIONS:
D/A Section
  Resolution: 10 bits
  Linearity: 0.2%
  Full Scale: 10.0V or ±5V
  Code: Offset binary
  Slew rate: 10V/μs
  Output impedance: 75 ohm at 20kHz
Comparator Section
  Number of channels: 8
  Max. offset voltage: 10mV
  Voltage range: ±10V
A/D Section
  Resolution: 10 bits
  Full Scale: 10V or ±5V
  Conversion Time: successive approximation 350μs max., tracking 50μs max. per channel
  Method: software controlled (drivers included for tracking and successive approximation conversion)

For further information contact:
POLYMORPHIC SYSTEMS
P.O. BOX 2207
GOLETA, CA 93018

Kit prices start at $135 for one D/A and $185 for two D/A's; bipolar option adds $8 for one D/A and $12 for two D/A's. This kit is available now and at these prices for these features, it should go over in a big way.

GROUNDING TECHNIQUES

There is probably no one in the field of electronics that has not had some experience with a noisy system. Perhaps you have a computer or instrumentation system that gives incorrect results or does inexplicable things. You could be the victim of noise. Many of these problems may be avoided by applying some basic principles of how to ground all interconnecting systems properly.

Electrical "ground" originally was a conducting connection between an electrical system and the earth; however, many times the term "ground" is incorrectly applied to a point used as a zero voltage reference. For example the common three wire 110 volt A.C. power receptacles required by the National Electrical Code have a grounded power conductor and a grounding conductor for safety. The grounded conductor is referred to as the neutral and is by definition only that current carrying conductor which is closest to true earth ground. The grounding wire is intended to be at earth ground and carry no load current. However, depending upon the quality and location of the ground connections, the neutral (white wire) may be several volts above true ground (green wire). This condition would be extreme, but several millivolts is a common power system ground loop voltage.

There are other types of grounds other than that of the power system. Signal grounds are the reference points and current return paths for all signal sources. Chassis and shield grounds are usually the circuitry's supporting hardware, case, and outer cable shields.

GROUND LOOPS

The fundamental principle is that each type of ground have the same potential throughout the system. This sounds ridiculously simple but is seldom completely fulfilled in any practical system. The problem is usually traceable to "ground loops" which is the term applied to current flowing in the ground path and, by virtue of the resistivity of the ground path medium, causes a voltage drop between two ground points.

Let's examine the primary ground loop. Low voltage facility distribution lines are all referenced to earth ground. This is done to minimize transient voltages due to arcing between grounds, to help stabilize the system, and to aid in protective fusing among other reasons.

Since all power systems are referenced to earth ground, currents are flowing in the earth. These currents find and flow along the paths of least resistance, producing potential drops in the earth as a result, and generating magnetic fluxes. These potential drops make it impossible to connect to the earth at two different points and have these points be at the same potential. The magnetic fluxes which are created couple through transformer action into long wires generating low impedance circulating currents which are difficult to eliminate.

Due to the extensive power distribution systems throughout the world, it must be assumed that ground currents exist almost everywhere. This is the 60 Hz signal that you see when you touch the open input of a scope or the hum you hear when the input of an amplifier is touched. Since this signal is so prevalent, it must be considered at all system interconnecting levels, especially if the systems are widely separated.

GROUND CONNECTIONS

There is no absolutely proper method for connecting together the grounds of computers, instruments, peripherals, etc. Depending upon your system configuration, there may be several ways which will yield satisfactory results. General guidelines for ground interconnections are:

1. Connect all grounds such that power and signal ground return currents cannot intermix but flow only in their own ground paths.
2. Keep all ground path connections as short as possible and use as large a conductor as practical between ground points to minimize the impedance between ground points.
3. Wherever possible avoid the use of multiple paths for ground currents.
4. The individual ground circuits *must* be designed such that high level ground currents cannot flow in low level input ground circuits.

The greatest obstacle to these criteria is the multiple ground connections that result from the three wire A.C. cord grounds of the different devices interconnected in a system. This can best be minimized through the use of large gage, bonding jumpers between chassis, to reduce any potential ground loop impedance.

ALTERNATE METHODS

A good compromise grounding system is to run the ground from the highest current level stage or device in a continuous line to the lowest current level stage or device in sequence with no doubling back (multiple paths). Even though a chassis is designated as ground, it cannot be over emphasized that the same potential does not exist at all points on the chassis. This is due to the higher specific resistivity of the steel or aluminum used for the chassis, as compared to the copper circuit wires. These points are even more important at higher frequencies.

Another good grounding method, known as single point grounding, involves running a completely separate wire from each device, power source, or signal source to only one physical ground point. This results in no common ground loop impedance except the output impedance of the common power source.

If your system is analog as well as digital in nature, connect the analog ground to the digital ground at only one physical point. Remember ground reference points are just that; points, not distributed parameters. Keep the grounding system simple since complex ground loops are an easy result to produce. ▣

Letters to the Editor

Dear Editor:

In the last issue we presented an Application Exchange noise cure that was submitted anonymously. At that time the validity of this particular ALTAIR modification had not been checked.

The modification in question involved the addition of various capacitors to buss control lines on the Display/Control board. (See page 17 of vol. 1 issue 1). The intent of the changes apparently was to reduce noise spikes by bypassing them to ground through the 'filter' capacitors.

Well, the modification worked too well as this author found out on his ALTAIR with the terminated Processor Technology Mother Board. Adding the caps turned a working 8800 into a computing idiot. Programs that ran fine before now wouldn't; although when the CPU was single stepped through a program everything appeared to be fine. Removing the capacitors returned the system to normal.

It is therefore recommended that no extra capacitance be added to any buss line signals on the ALTAIR 8800. The apparent cause of failure was due to disruption of the timing relationships of the buss lines involved. (The caps and the buss terminating resistors make dandy R/C pulse stretching timing networks). Our apologies for any inconvenience caused.

S. Wilcox

FIRST HONG KONG MEMBER

SCCS welcomes Mr. John Ng as our first member in Hong Kong. Mr. Ng writes:

Dear Editor:

I am overjoyed to be invited to be a club member. I have filled in the membership form enclosed with this letter together with a bankdraft of sum $15 to cover membership fee and additional mailing charges for the time being.

At present I am trying hard in setting up an ALTAIR 8800 8K Basic system—but I haven't got a Teletype and I am expecting some difficulty in interfacing a TV monitor to it. I look forward to getting valuable tips and help from SCCS & INTERFACE.

Mr. Ng Wing Yu (John Ng)
96, Pokfulam Road
A2, 3/F, Hong Kong.

Ed.

Perhaps some of the SCCS members would like to write Mr. Ng directly to offer assistance and to discover if computing in Hong Kong is as much fun as it is in the U.S.A. (Anyone for Chinese-to-English translation program?)

Additional insight can be gained by obtaining documentation on the TVT-2 keyboard and display unit from Southwest Technical Products, San Antonio, Texas. (You may have to account for differences that may exist in the horizontal and vertical oscillators between U.S. and British T.V. systems).

"BYTE MAGAZINE"—Green Publishing Co. has carried some articles on T.V. display systems that may also be applicable.

We would like to assist you further, but need more specific direction as to your T.V. monitor problems.

Good luck and welcome aboard!

Dear Editor:

After hearing the discussions and presentations at the last two S.C.C.S. meetings for a hobbyist tape standard, I am concerned that nothing is being said of software format standard; a standard that, of necessity, must also be agreed upon. I agree with the editors of BYTE magazine that a software vacuum exists in the hobbyist computer field. It seems that the field is so new that most of us are still struggling financially to build our systems up, assuming that we can take care of the necessary programs later. This may be so.

For the immediate situation every hobbyist has a need to read and write programs or data from and to tape. Most of us have developed simple dump type programs to do this. But for meaningful exchanges of program material some decision must be made by the majority of the hobbyists. I think that the Southern California Computer Society has some of the power to influence this decision process.

The "Popular Electronics" HIT format is OK but has some drawbacks I feel. Its major fault is that it has no provision for a memory reference address pointer in the preamble of the data record format. This is necessary to know where in memory the program (or data) is designed to operate from without changing all referenced addresses in the program. Also what happens when the day comes that you have more program material developed than you have memory to dump it into. The advantage of tape as a memory medium is unlimited capacity. I prefer to build a program library on tape with each program "named" with its starting address. This will allow for the use of a tape search program whereby you may name the desired program to be loaded to memory from tape and simply play the entire tape and not have to worry exactly where on the tape it is recorded.

As for form of text, I see only two really viable possibilities for the hobbyist. The simplest solution is to exchange program material directly in machine code. Which is fine except that without extra work you are limited to running programs written to run on the CPU chip that your system is based upon. And a higher level language is not the answer either; since a BASIC, for example, written to run on an 8080 based system uses 8080 machine code and cannot run directly on a 6800 MPU based system.

A more encompassing approach would be the development, by S.C.C.S. maybe, of a public domain library of cross assembler programs to translate 8008 programs to 6800, or LSI-11 to PACE, or 8080 to 6800 to PDP-8 to etc.

I believe this whole subject warrents more discussion and feedback from the membership. However, in the meantime I offer the following modified version of the HIT data record format. (octal representation)

1st word on tape section	026	ASCII 'SYN' word recorded minimum 32 times (for AGC settling)
33rd word on tape section	XXX	Lo memory ref. pointer
	XXX	Hi memory ref. pointer
	XXX	Lo byte No. of words
	XXX	Hi byte following 'STX'
37th word on tape section	002	ASCII 'STX' start of text
	+	
	+	
	+	Text
	+	
	+	
	+	
	+	
	003	ASCII 'ETX' end of text
	XXX	Lo byte check sum
Last word on tape section	XXX	Hi byte check sum

The check sum in this case would be the overflow bit sum for the first word through 'ETX' word.

S. Wilcox

Ed.

Our thanks to Scott for broaching an important subject—one that is going to require considerable discussion, thought and creative effort to make the world of hobby computing as pleasant as we all want it to be. Responses to Scott's proposal and suggestions for solutions to the software standardization problem are invited. (See Larry Press' proposal for group design projects regarding the idea of a library of cross compilers.)

SPECIAL NOTE: The following are excerpts taken from a letter written by Larry Press. The idea and projects proposed by Larry are worthy of serious consideration by members of SCCS.

GROUP DESIGN PROJECT I—DIAGNOSTICS

Let's try to find innovative ways to use INTERFACE, not only as a forum for a few writers "broadcasting" information to a large, passive audience, but as a device to stimulate interaction between the members as well. One thing we might try is the initiation of group design projects. If someone has interest in some sort of hardware or software and feels that there is a need for it, he could try to "convene" a design group by writing a description of the problem and calling for interested collaborators to contact him.

Each such project would evolve in its own way; however, all would probably involve establishing contact between the people who respond to the initial call for collaborators and the publication of results of their work in future issues of INTERFACE. The publication of results might be broken into two or more installments, corresponding to design iterations. For example a preliminary publication of external specifications of the device or program followed by an article on the results of the implementation effort.

Think about it and see if you wouldn't like to try initiating group effort to design some hardware or software. Let me offer the first attempt—I'm planning to solder together an ALTAIR CPU. Since the most sophisticated piece of electronic test equipment I've ever used is the tube tester in a Savon Drug Store, I'm expecting that my ALTAIR won't work when first plugged in. Not only that, I suspect that it might even require a little maintenance from time to time.

Therefore, I would like to initiate a group effort to specify and then implement, a package of diagnostic programs for the ALTAIR 8800. If you have interest in or knowledge of diagnostic programming, let me hear from you. I'd like references to good literature, names of people with 8800 diagnostics, preliminary versions of external specifications, etc.

The next step will be to establish communication among the people who respond to this call for collaborators. They might put together an INTERFACE article surveying the initial responses and proposing external specifications for a set of diagnostics. The next iteration might produce a suite of programs for the 8800.

If necessary, I am willing to coordinate the responses myself; however, in view of my limited qualifications, I would prefer that someone else be coordinator. When you respond, let me know if you would be interested in taking over that role.

Ed.

I know of some work that has already been done in this area (see Jon Waldon's memory test program on page 31) and I'm sure there is much more. Those interested may write Larry directly at 128 Park Place, Venice, Calif. 90291, or contact him through this magazine.

Dear Editor:

A couple more ideas the members might care to consider are useful projects that your editor would like to do, but simply hasn't had the time after putting out this magazine each month and doing 40+ a week to feed the face.

The first, now in progress, is a software project—a full blown trace program for use in debugging 8080 assembly language programs.

The decision to start the project was prompted by the apparent naive statement in the MITS Altair 8800 Operator's Manual. To quote from page 41, paragraph 3, "Occasionally it will be necessary to 'debug' a program. The need for debugging occurs when a program fails to execute properly because of errors (bugs). Debugging can be enhanced by use of the Single Step switch."

A novice computer hobbyist is going to be in for quite a shock if he believes the word "occasionally", particularly after he graduates from the simple "cute routines" stage to the complexity of more useful programs. Further, are there really any experienced assembly language programmers out there who "single step" through a complex program? If so, you probably wear out thumbs and switches with amazing regularity and will soon look for a better way.

That better way is a program which executes the program under test, instruction by instruction, displaying as it proceeds, the address of the instruction, the instruction itself, the contents of all registers and any memory locations referenced by the instructions, the relevant portion of the stacks and the program status word. The program also provides the facility to start and stop execution at any address, start and stop the display function at any address, and alter the contents of memory.

The other idea is a color TV graphics project requiring considerable hardware expertise as well as software talent—one that might serve a humanitarian purpose. It will be covered in a separate article.

Thanks for the good suggestion, Larry, and for the stimulation. All you other guys—start designing. SCCS INTERFACE will lend all the assistance possible.

Art Childs

APPLICATIONS EXCHANGE

ALTAIR 8800 MEMORY CHECK PROGRAM

It seems that many of the members of S.C.C.S. are owners or soon to be owners of ALTAIR'S (myself included) and are in the first stages of going through the learning curve(s) of this "new" microcomputer technology. Building the hardware is only a small part of this hobby, because once everything is soldered together you're done. Software is where the real challenge and excitement come in. There is no "done" in software where you may sit back and say "I've done all the programs". There is no such point, as software can keep you occupied for years (I hope not on the same program). So to those members still in their learning curve I offer two basic programs for ALTAIR owners.

The first is a memory check program, of which there are several running around. This one is useful when you've just finished that new 4k board you bought from so-n-so and you're wondering if it's OK (memory chips do have a *small* failure rate).

The second is a cassette tape read and write program so you can dump all those programs you were playing with before you put your 8800 to bed and it forgets all it learned. This program was written to operate through the MITS Inc. Audio Cassette Interface, although any I/O parallel to serial modem operating around 2kHz should work. Any medium quality recorder can be used.

The tape I/O port is addressed for control channel - 6 and data channel - 7. Put the bootstrap in manually; it will load the tape input program, which must be the first program recorded. Make sure that the starting address of where the bootstrap is to start loading into memory is four addresses lower than where the body of the program being loaded is to start. This is because the boot is a dump type program and will load into memory the 4 preamble words that the output program places before each program it writes. After the programs are initially entered the output program can be used to write itself onto tape.

SCOTT WILCOX
405-B No. Monterey
Alhambra, CA 91801

DESCRIPTION: This program checks any block of directly addressable memory for read/write data errors. All possible bit patterns are read in and out of each memory location. The program stops (enters a futile loop) and stores the address of the "bad" memory location and the data byte that execution failed on for three situations:

1. a faulty memory location was encountered (data read did not equal data written).
2. a protected block of memory was encountered.
3. the program "stop memory check" address was set higher than the amount of memory actually in the machine.

```
ADDRESS      MNEMONIC              COMMENT
000          LXI H,L        041    Load stop address of memory check  ( Address of last mem +1 )
001                         XXX  377
002                         XXX  001
003          LXI D,E        021    Load start address of memory check
004                         XXX  057
005                         XXX  [illegible]
006          MVI A          076
007                         377    Initial check data
010          MOV A,B        107    Save data for comparison to memory read data
011          STAX D,E       022
012          STA            062    Store data in case memory fails
013                         056
014                         000
015          LDAX D,E       032    Read data for comparison
016          CMP B          270    compare read/write data
017          JZ             312    Loop if no memory error
020                         031
021                         000
```

```
022      XCHG      353
023      SHLD      042       Store address of bad memory
024                054
025                000
026      JMP       303       end of program futile loop
027                023
030                000
031      SUI       326       decrement check data
032                001
033      JNC       322       loop if not final data case (000)
034                010
035                000
036      INX D,E   023       increment to next memory address
037      MOV D,A   172
040      CMP H     274
041      JNZ       302       loop if not last memory check address
042                006
043                000
044      MOV E,A   173
045      CMP L     275
046      JNZ       302       loop if not last memory check address
047                006
050                000
051      JMP       303       loop to store stop address
052                023
053                000
054                XXX       lo address of bad memory
055                XXX       hi address of bad memory
056                XXX       data case that memory failed on
```

LIMITATIONS: The program requires 46 bytes of known good memory and must not be located in a protected block of memory. Running time is approximately 30 sec. for each 4k of memory being checked.

TAPE INPUT (PLAYBACK) BOOTSTRAP LOADER

```
ADDRESS              DATA
000000   LXI H,L     041
000001               XXX   (L)   Start address of program to be loaded into memory
000002               XXX   (H)
000003   IN 6        333         Control channel input
000004               006
000005   RRC         017         Check for input data ready
000006   JC          332         Loop if not ready
000007               003   (L)
000010               000   (H)
000011   IN 7        333         Data channel input
000012               007
000013   MOV M,A     167         Move input data from accumulator to memory
000014   INX H,L     043         Increment memory reference address
000015   JMP         303         Loop for next input data byte
000016               003
000017               000
```

*Bootstrap loads TAPE INPUT PROGRAM from the *start* of the tape leader

TAPE INPUT PROGRAM (PLAYBACK)

```
000100   LXI SP      061
000101               300   (L)   Set stack memory location
000102               000   (H)
```

```
000103      CALL "INPUT"   315
000104                     150   (L)
000105                     000   (H)
000106      CPI            376         Compare input for first check word
000107                     002         "STX" (start of text)
000110      JNZ            302         Loop if not check word
000111                     103   (L)
000112                     000   (H)
000113      CALL "INPUT"   315
000114                     150   (L)
000115                     000   (H)
000116      CPI            376         Compare input for second check word
000117                     002         "STX" (start of text)
000120      JNZ            302         Loop if not check word
000121                     103   (L)
000122                     000   (H)
000123      CALL "INPUT"   315
000124                     150   (L)
000125                     000   (H)
000126      MOV L,A        157         Load low address of memory pointer
000127      CALL "INPUT"   315
000130                     150   (L)
000131                     000   (H)
000132      MOV H,A        147         Load high address of memory pointer
000133      CALL "INPUT"   315
000134                     150   (L)
000135                     000   (H)
000136      MOV M,A        167         Load memory reference data to accumulator
000137      INX H,L        043         Increment memory reference address
000140      JMP            303         Loop for next input data byte
000141                     133   (L)
000142                     000   (H)
"INPUT"
000150      STC            037         Set carry bit
000151      IN 6           333         Input control channel
000152                     006
000153      RRC            017         Check for input data ready
000154      JC             332         Loop if not ready
000155                     150   (L)
000156                     000   (H)
000157      IN 7           333         Input data channel
000160                     007
000161      RET            311         Return to main program
```

TAPE OUTPUT PROGRAM (RECORD)

```
ADDRESS                  DATA
000200      LXI SP         061         Set stack memory location
000201                     300   (L)
000202                     000   (H)
000203      LXI H,L        041         Set start of memory block to be recorded
000204                     XXX   (L)
000205                     XXX   (H)
000206      LXI D,E        021         Set stop address of memory block to be recorded
000207                     XXX   (L)
000210                     XXX   (H)
000211      MVI B          006         Load register B with first check word to be outputted
000212                     002         "STX" (start of text)
```

```
000213      CALL "OUT"    315
000214                    260  (L)
000215                    000  (H)
000216      MVI B         006        Load register B with 2nd check word to be recorded
000217                    002        "STX" (start of text)
000220      CALL "OUT"    315
000221                    260  (L)
000222                    000  (H)
000223      MOV B,L       105        Record low memory reference address pointer
000224      CALL "OUT"    315
000225                    260  (L)
000226                    000  (H)
000227      MOV B,H       104        Record high memory reference address pointer
000230      CALL "OUT"    315
000231                    260  (L)
000232                    000  (H)
000233      MOV B,M       106        Record data at memory reference address
000234      CALL "OUT"    315
000235                    260  (L)
000236                    000  (H)
000237      INX H,L       043        Increment memory reference address
000240      MOV A,H       174
000241      CMP D         272        Check for stop address
000242      JNZ           302        Loop if not stop address
000243                    233  (L)
000244                    000  (H)
000245      MOV A,L       175
000246      CMP E         273        Check for stop address
000247      JNZ           302        Loop if not stop address
000250                    233  (L)
000251                    000  (H)
000252      JMP           303        Futile loop end of program
000253                    247  (L)
000254                    000  (H)
"OUT"
000260      IN 6          333        Control channel input
000261                    006
000262      RLC           007        Check output ready to accept data
000263      JC            332        Loop if not ready
000264                    260  (L)
000265                    000  (H)
000266      MOV A,B       170        Move data from holding register to accumulator
000267      OUT 7         323        Output data channel
000270                    007
000271      RET           311        Return to main program
```

Tape preamble leader format for each memory block recorded:

byte 1	first start of transmission check word ASCII "STX", 002 octal
byte 2	second check word (repeat of "STX")
byte 3	low address pointer where following program is to be written
byte 4	high address pointer where following program is to be written

*Preamble is decoded by input program but not written into memory

8080 INSTRUCTION SET: NUMERICAL

NUL	0	00	000		NOP		@	64	40	100	MOV B→B	128	80	200	ADD B	α	192	C0	300		RNZ	
SOH	1	01	001	≡	LXI B		A	65	41	101	MOV C→B	129	81	201	ADD C	α	193	C1	301		POP B	
STX	2	02	002		STAX B		B	66	42	102	MOV D→B	130	82	202	ADD D	α	194	C2	302	≡	JNZ	
ETX	3	03	003		INX B		C	67	43	103	MOV E→B	131	83	203	ADD E	α	195	C3	303	≡	JMP	
EOT	4	04	004		INR B	β	D	68	44	104	MOV H→B	132	84	204	ADD H	α	196	C4	304	≡	CNZ	
ENQ	5	05	005		DCR B	β	E	69	45	105	MOV L→B	133	85	205	ADD L	α	197	C5	305		PUSH B	
ACK	6	06	006	=	MVI B		F	70	46	106	MOV M→B	134	86	206	ADD M	α	198	C6	306	=	ADI	α
BEL	7	07	007		RLC	γ	G	71	47	107	MOV A→B	135	87	207	ADD A	α	199	C7	307		RST 000	
BS	8	08	010		--		H	72	48	110	MOV B→C	136	88	210	ADC B	α	200	C8	310		RZ	
HT	9	09	011		DAD B	γ	I	73	49	111	MOV C→C	137	89	211	ADC C	α	201	C9	311		RET	
LF	10	0A	012		LDAX B		J	74	4A	112	MOV D→C	138	8A	212	ADC D	α	202	CA	312	≡	JZ	
VT	11	0B	013		DCX B		K	75	4B	113	MOV E→C	139	8B	213	ADC E	α	203	CB	313		--	
FF	12	0C	014		INR C	β	L	76	4C	114	MOV H→C	140	8C	214	ADC H	α	204	CC	314	≡	CZ	
CR	13	0D	015		DCR C	β	M	77	4D	115	MOV L→C	141	8D	215	ADC L	α	205	CD	315	≡	CALL	
SO	14	0E	016	=	MVI C		N	78	4E	116	MOV M→C	142	8E	216	ADC M	α	206	CE	316	=	ACI	α
SI	15	0F	017		RRC	γ	O	79	4F	117	MOV A→C	143	8F	217	ADC A	α	207	CF	317		RST 010	
DLE	16	10	020		--		P	80	50	120	MOV B→D	144	90	220	SUB B	α	208	D0	320		RNC	
DC1	17	11	021	≡	LXI D		Q	81	51	121	MOV C→D	145	91	221	SUB C	α	209	D1	321		POP D	
DC2	18	12	022		STAX D		R	82	52	122	MOV D→D	146	92	222	SUB D	α	210	D2	322	≡	JNC	
DC3	19	13	023		INX D		S	83	53	123	MOV E→D	147	93	223	SUB E	α	211	D3	323	=	OUT	
DC4	20	14	024		INR D	β	T	84	54	124	MOV H→D	148	94	224	SUB H	α	212	D4	324	≡	CNC	
NAK	21	15	025		DCR D	β	U	85	55	125	MOV L→D	149	95	225	SUB L	α	213	D5	325		PUSH D	
SYN	22	16	026	=	MVI D		V	86	56	126	MOV M→D	150	96	226	SUB M	α	214	D6	326	=	SUI	α
ETB	23	17	027		RAL	γ	W	87	57	127	MOV A→D	151	97	227	SUB A	α	215	D7	327		RST 020	
CAN	24	18	030		--		X	88	58	130	MOV B→E	152	98	230	SBB B	α	216	D8	330		RC	
EM	25	19	031		DAD D	γ	Y	89	59	131	MOV C→E	153	99	231	SBB C	α	217	D9	331		--	
SUB	26	1A	032		LDAX D		Z	90	5A	132	MOV D→E	154	9A	232	SBB D	α	218	DA	332	≡	JC	
ESC	27	1B	033		DCX D		[	91	5B	133	MOV E→E	155	9B	233	SBB E	α	219	DB	333	=	IN	
FS	28	1C	034		INR E	β	\	92	5C	134	MOV H→E	156	9C	234	SBB H	α	220	DC	334	≡	CC	
GS	29	1D	035		DCR E	β	]	93	5D	135	MOV L→E	157	9D	235	SBB L	α	221	DD	335		--	
RS	30	1E	036	=	MVI E		^	94	5E	136	MOV M→E	158	9E	236	SBB M	α	222	DE	336	=	SBI	α
US	31	1F	037		RAR	γ	_	95	5F	137	MOV A→E	159	9F	237	SBB A	α	223	DF	337		RST 030	
SP	32	20	040		--		`	96	60	140	MOV B→H	160	A0	240	ANA B	α	224	E0	340		RPO	
!	33	21	041	≡	LXI H		a	97	61	141	MOV C→H	161	A1	241	ANA C	α	225	E1	341		POP H	
"	34	22	042	≡	SHLD		b	98	62	142	MOV D→H	162	A2	242	ANA D	α	226	E2	342	≡	JPO	
#	35	23	043		INX H		c	99	63	143	MOV E→H	163	A3	243	ANA E	α	227	E3	343		XTHL	
$	36	24	044		INR H	β	d	100	64	144	MOV H→H	164	A4	244	ANA H	α	228	E4	344	≡	CPO	
%	37	25	045		DCR H	β	e	101	65	145	MOV L→H	165	A5	245	ANA L	α	229	E5	345		PUSH H	
&	38	26	046	=	MVI H		f	102	66	146	MOV M→H	166	A6	246	ANA M	α	230	E6	346	=	ANI	α
'	39	27	047		DAA	α	g	103	67	147	MOV A→H	167	A7	247	ANA A	α	231	E7	347		RST 040	
(	40	28	050		--		h	104	68	150	MOV B→L	168	A8	250	XRA B	α	232	E8	350		RPE	
)	41	29	051		DAD H	γ	i	105	69	151	MOV C→L	169	A9	251	XRA C	α	233	E9	351		PCHL	
*	42	2A	052	≡	LHLD		j	106	6A	152	MOV D→L	170	AA	252	XRA D	α	234	EA	352	≡	JPE	
+	43	2B	053		DCX H		k	107	6B	153	MOV E→L	171	AB	253	XRA E	α	235	EB	353		XCHG	
,	44	2C	054		INR L	β	l	108	6C	154	MOV H→L	172	AC	254	XRA H	α	236	EC	354	≡	CPE	
-	45	2D	055		DCR L	β	m	109	6D	155	MOV L→L	173	AD	255	XRA L	α	237	ED	355		--	
.	46	2E	056	=	MVI L		n	110	6E	156	MOV M→L	174	AE	256	XRA M	α	238	EE	356	=	XRI	α
/	47	2F	057		CMA		o	111	6F	157	MOV A→L	175	AF	257	XRA A	α	239	EF	357		RST 050	
0	48	30	060		--		p	112	70	160	MOV B→M	176	B0	260	ORA B	α	240	F0	360		RP	
1	49	31	061	≡	LXI SP		q	113	71	161	MOV C→M	177	B1	261	ORA C	α	241	F1	361		POP PSW	α
2	50	32	062	≡	STA		r	114	72	162	MOV D→M	178	B2	262	ORA D	α	242	F2	362	≡	JP	
3	51	33	063		INX SP		s	115	73	163	MOV E→M	179	B3	263	ORA E	α	243	F3	363		DI	
4	52	34	064		INR M	β	t	116	74	164	MOV H→M	180	B4	264	ORA H	α	244	F4	364	≡	CP	
5	53	35	065		DCR M	β	u	117	75	165	MOV L→M	181	B5	265	ORA L	α	245	F5	365		PUSH PSW	
6	54	36	066	=	MVI M		v	118	76	166	HLT	182	B6	266	ORA M	α	246	F6	366	=	ORI	α
7	55	37	067		STC	γ	w	119	77	167	MOV A→M	183	B7	267	ORA A	α	247	F7	367		RST 060	
8	56	38	070		--		x	120	78	170	MOV B→A	184	B8	270	CMP B	α	248	F8	370		RM	
9	57	39	071		DAD SP	γ	y	121	79	171	MOV C→A	185	B9	271	CMP C	α	249	F9	371		SPHL	
:	58	3A	072	≡	LDA		z	122	7A	172	MOV D→A	186	BA	272	CMP D	α	250	FA	372	≡	JM	
;	59	3B	073		DCX SP		{	123	7B	173	MOV E→A	187	BB	273	CMP E	α	251	FB	373		EI	
<	60	3C	074		INR A	β	¦	124	7C	174	MOV H→A	188	BC	274	CMP H	α	252	FC	374	≡	CM	
=	61	3D	075		DCR A	β	}	125	7D	175	MOV L→A	189	BD	275	CMP L	α	253	FD	375		--	
>	62	3E	076	=	MVI A		~	126	7E	176	MOV M→A	190	BE	276	CMP M	α	254	FE	376	=	CPI	α
?	63	3F	077		CMC	γ	DEL	127	7F	177	MOV A→A	191	BF	277	CMP A	α	255	FF	377		RST 070	

8080 INSTRUCTION SET: ALPHABETICAL

α	ACI	=	316	CE	206
α	ADC A		217	8F	143
α	ADC B		210	88	136
α	ADC C		211	89	137
α	ADC D		212	8A	138
α	ADC E		213	8B	139
α	ADC H		214	8C	140
α	ADC L		215	8D	141
α	ADC M		216	8E	142
α	ADD A		207	87	135
α	ADD B		200	80	128
α	ADD C		201	81	129
α	ADD D		202	82	130
α	ADD E		203	83	131
α	ADD H		204	84	132
α	ADD L		205	85	133
α	ADD M		206	86	134
α	ADI	=	306	C6	198
α	ANA A		247	A7	167
α	ANA B		240	A0	160
α	ANA C		241	A1	161
α	ANA D		242	A2	162
α	ANA E		243	A3	163
α	ANA H		244	A4	164
α	ANA L		245	A5	165
α	ANA M		246	A6	166
α	ANI	=	346	E6	230
	CALL	≡	315	CD	205
	CC	≡	334	DC	220
	CM	≡	374	FC	252
	CMA		057	2F	47
γ	CMC		077	3F	63
α	CMP A		277	BF	191
α	CMP B		270	B8	184
α	CMP C		271	B9	185
α	CMP D		272	BA	186
α	CMP E		273	BB	187
α	CMP H		274	BC	188
α	CMP L		275	BD	189
α	CMP M		276	BE	190
	CNC	≡	324	D4	212
	CNZ	≡	304	C4	196
	CP	≡	364	F4	244
	CPE	≡	354	EC	236
α	CPI	=	376	FE	254
	CPO	≡	344	E4	228
	CZ	≡	314	CC	204
α	DAA		047	27	39
γ	DAD B		011	09	9
γ	DAD D		031	19	25
γ	DAD H		051	29	41
γ	DAD SP		071	39	57
β	DCR A		075	3D	61
β	DCR B		005	05	5
β	DCR C		015	0D	13
β	DCR D		025	15	21
β	DCR E		035	1D	29
β	DCR H		045	25	37
β	DCR L		055	2D	45
β	DCR M		065	35	53
	DCX B		013	0B	11
	DCX D		033	1B	27
	DCX H		053	2B	43
	DCX SP		073	3B	59
	DI		363	F3	243
	EI		373	FB	251
	HLT		166	76	118
	IN	=	333	DB	219
β	INR A		074	3C	60
β	INR B		004	04	4
β	INR C		014	0C	12
β	INR D		024	14	20
β	INR E		034	1C	28
β	INR H		044	24	36
β	INR L		054	2C	44
β	INR M		064	34	52
	INX B		003	03	3
	INX D		023	13	19
	INX H		043	23	35
	INX SP		063	33	51
	JC	≡	332	DA	218
	JM	≡	372	FA	250
	JMP	≡	303	C3	195
	JNC	≡	322	D2	210
	JNZ	≡	302	C2	194
	JP	≡	362	F2	242
	JPE	≡	352	EA	234
	JPO	≡	342	E2	226
	JZ	≡	312	CA	202
	LDA	≡	072	3A	58
	LDAX B		012	0A	10
	LDAX D		032	1A	26
	LHLD	≡	052	2A	42
	LXI B	≡	001	01	1
	LXI D	≡	021	11	17
	LXI H	≡	041	21	33
	LXI SP	≡	061	31	49
	MOV A→A		177	7F	127
	MOV A→B		107	47	71
	MOV A→C		117	4F	79
	MOV A→D		127	57	87
	MOV A→E		137	5F	95
	MOV A→H		147	67	103
	MOV A→L		157	6F	111
	MOV A→M		167	77	119
	MOV B→A		170	78	120
	MOV B→B		100	40	64
	MOV B→C		110	48	72
	MOV B→D		120	50	80
	MOV B→E		130	58	88
	MOV B→H		140	60	96
	MOV B→L		150	68	104
	MOV B→M		160	70	112
	MOV C→A		171	79	121
	MOV C→B		101	41	65
	MOV C→C		111	49	73
	MOV C→D		121	51	81
	MOV C→E		131	59	89
	MOV C→H		141	61	97
	MOV C→L		151	69	105
	MOV C→M		161	71	113
	MOV D→A		172	7A	122
	MOV D→B		102	42	66
	MOV D→C		112	4A	74
	MOV D→D		122	52	82
	MOV D→E		132	5A	90
	MOV D→H		142	62	98
	MOV D→L		152	6A	106
	MOV D→M		162	72	114
	MOV E→A		173	7B	123
	MOV E→B		103	43	67
	MOV E→C		113	4B	75
	MOV E→D		123	53	83
	MOV E→E		133	5B	91
	MOV E→H		143	63	99
	MOV E→L		153	6B	107
	MOV E→M		163	73	115
	MOV H→A		174	7C	124
	MOV H→B		104	44	68
	MOV H→C		114	4C	76
	MOV H→D		124	54	84
	MOV H→E		134	5C	92
	MOV H→H		144	64	100
	MOV H→L		154	6C	108
	MOV H→M		164	74	116
	MOV L→A		175	7D	125
	MOV L→B		105	45	69
	MOV L→C		115	4D	77
	MOV L→D		125	55	85
	MOV L→E		135	5D	93
	MOV L→H		145	65	101
	MOV L→L		155	6D	109
	MOV L→M		165	75	117
	MOV M→A		176	7E	126
	MOV M→B		106	46	70
	MOV M→C		116	4E	78
	MOV M→D		126	56	86
	MOV M→E		136	5E	94
	MOV M→H		146	66	102
	MOV M→L		156	6E	110
	MVI A	=	076	3E	62
	MVI B	=	006	06	6
	MVI C	=	016	0E	14
	MVI D	=	026	16	22
	MVI E	=	036	1E	30
	MVI H	=	046	26	38
	MVI L	=	056	2E	46
	MVI M	=	066	36	54
	NOP		000	00	0
α	ORA A		267	B7	183
α	ORA B		260	B0	176
α	ORA C		261	B1	177
α	ORA D		262	B2	178
α	ORA E		263	B3	179
α	ORA H		264	B4	180
α	ORA L		265	B5	181
α	ORA M		266	B6	182
α	ORI	=	366	F6	246
	OUT	=	323	D3	211
	PCHL		351	E9	233
	POP B		301	C1	193
	POP D		321	D1	209
	POP H		341	E1	225
α	POP PSW		361	F1	241
	PUSH B		305	C5	197
	PUSH D		325	D5	213
	PUSH H		345	E5	229
	PUSH PSW		365	F5	245
γ	RAL		027	17	23
γ	RAR		037	1F	31
	RC		330	D8	216
	RET		311	C9	201
γ	RLC		007	07	7
	RM		370	F8	248
	RNC		320	D0	208
	RNZ		300	C0	192
	RP		360	F0	240
	RPE		350	E8	232
	RPO		340	E0	224
γ	RRC		017	0F	15
	RST 000		307	C7	199
	RST 010		317	CF	207
	RST 020		327	D7	215
	RST 030		337	DF	223
	RST 040		347	E7	231
	RST 050		357	EF	239
	RST 060		367	F7	247
	RST 070		377	FF	255
	RZ		310	C8	200
α	SBB A		237	9F	159
α	SBB B		230	98	152
α	SBB C		231	99	153
α	SBB D		232	9A	154
α	SBB E		233	9B	155
α	SBB H		234	9C	156
α	SBB L		235	9D	157
α	SBB M		236	9E	158
α	SBI	=	336	DE	222
	SHLD	≡	042	22	34
	SPHL		371	F9	249
	STA	≡	062	32	50
γ	STC		067	37	55
	STAX B		002	02	2
	STAX D		022	12	18
α	SUB A		227	97	151
α	SUB B		220	90	144
α	SUB C		221	91	145
α	SUB D		222	92	146
α	SUB E		223	93	147
α	SUB H		224	94	148
α	SUB L		225	95	149
α	SUB M		226	96	150
α	SUI	=	326	D6	214
	XCHG		353	EB	235
α	XRA A		257	AF	175
α	XRA B		250	A8	168
α	XRA C		251	A9	169
α	XRA D		252	AA	170
α	XRA E		253	AB	171
α	XRA H		254	AC	172
α	XRA L		255	AD	173
α	XRA M		256	AE	174
α	XRI	=	356	EE	238
	XTHL		343	E3	227

The instruction sets on pages 49 and 50 are designed to fit on one page each. The coding explanations which follow are left off since the pages are filled to capacity. The alphabetical set on page 50 reads:

Flag symbol, Mnemonic, Byte symbol, Octal, Hex, Decimal . . . etc. . . .

The numeric set on page 49 reads:

7-bit ASCII (1st two columns only), Decimal, Hex, Octal, Byte symbol, Mnemonic, Flag symbol . . . etc. . . .

Flag symbols are:	a	means all flags affected
	B	means all flags except CARRY affected
	y	means CARRY only affected
	none	means no flags affected
Byte symbols are:	=	means a 3-byte instruction
	=	means a 2-byte instruction
	none	means a 1-byte instruction

You might want to note that the non-printing 7-bit ASCII codes (and some printing codes) vary with different keyboards and operating systems. For example, Processor Technology's system uses the following variations:

HEX	ASCII	HEX	ASCII
09	TAB	5E	
OC	FORM	5F	
11	X-ON	7D	ALT MODE
12	TAPE	7F	RUB OUT
13	X-OFF		

A particular machine may substitute F1 for ESC or INT for ENQ. Check out the ASCII for any special keys you may have and make note of the special uses a system may make of Ox and 1x (control) codes.

CULTURE FOR COMPUTERS

by John Whitney

We are all a bunch of savages! Or so John R. Pierce would have us realize as regards our musical talents with computers. Since the computer can generate any waveform, we can produce any musical sound ever heard by man, past or future. Still we are like the innocent savage confronted with a Steinway concert grand. That's how poorly we know how to "play" those computer generated wave forms.

The same state of innocence holds true with regard to our "talents" with computer graphics. Briefly stated regarding the role of computer graphics as a newcomer in the visual arts field, this writer and IT are newcomers. Indeed, aren't we all newcomers and quite savage about it?

The computer's best graphic capability is as a plotter of mathematical dynamics. Never before was there instrumentality to compare with this computer instrument for the visualization of numbers in motion. We have Descartes to thank for making it possible to visualize the subtleties of number function by way of the X and Y coordinates of a plotted graph. We have the computer to thank for turning all his illuminations into a real-time dynamic and alive experience. And we may rejoice that real-time alive experiences usually have some intrinsic aesthetic merit. Aesthetic experience has much to do with that which is alive and vital. Eureka!! We have found a new instrument for visual art—a visual art with unique and special dimensions in time and motion.

For there is one good reason (among many) that tells us why computer graphics have such a grand potential for art: DYNAMICS.

Computers are known for their capability to crunch numbers ceaselessly. Punch in some number functions with some geometry and plug in a CRT, and you're likely to produce a fairly interesting abstract pattern display. But if you play with numbers and functions imaginatively, the chances are that you will produce a series of displays which, when viewed as a 24-frame-per-second sequence, you may find it more interesting for the quality of motion than for the pattern of any individual frame.

Possibilities for something quite new lay in this unique character of the computer. It will produce hundreds of thousands of slightly incremented displays as easily as it can produce one. We can state the special opportunity to be found in all this another way: for a newcomer, we got the goddamndest movie machine ever, and it'll make music as well! We better get on the with task of learning how to "play" it.

More about this in future issues.

APPLICATIONS EXCHANGE

64K MEMORY CHART

The chart below identifies (in hex and octal) the high order portion of the address for any given 1/4K of memory. Each 1/4K begins with low order address 00_{16} (000_8) and ends with FF_{16} (377_8). Each K is sectioned off for easy identification. Examples:

Section:	Begins with:		Ends with:	
	HX	OCT	HX	OCT
The 8th K	1C00	034 000	1FFF	037 377
The 3rd quarter of all 64K	8000	200 000	BFFF	277 377
The 2nd quarter of the 2nd K	0600	006 000	06FF	006 377
The 38th & 39th K	9400	224 000	9BFF	233 377

HX	OCT	K	HX	OCT	K	HX	OCT	K	HX	OCT	K	HX	OCT	K	HX	OCT	K	HX	OCT	K	HX	OCT	K
00	000	1	20	040	1	40	100	1	60	140	1	80	200	1	A0	240	1	C0	300	1	E0	340	1
01	001	2	21	041	2	41	101	2	61	141	2	81	201	2	A1	241	2	C1	301	2	E1	341	2
02	002	3	22	042	3	42	102	3	62	142	3	82	202	3	A2	242	3	C2	302	3	E2	342	3
03	003	1	23	043	9	43	103	17	63	143	25	83	203	33	A3	243	41	C3	303	49	E3	343	57
04	004	1	24	044	1	44	104	1	64	144	1	84	204	1	A4	244	1	C4	304	1	E4	344	1
05	005	2	25	045	2	45	105	2	65	145	2	85	205	2	A5	245	2	C5	305	2	E5	345	2
06	006	3	26	046	3	46	106	3	66	146	3	86	206	3	A6	246	3	C6	306	3	E6	346	3
07	007	2	27	047	10	47	107	18	67	147	26	87	207	34	A7	247	42	C7	307	50	E7	347	58
08	010	1	28	050	1	48	110	1	68	150	1	88	210	1	A8	250	1	C8	310	1	E8	350	1
09	011	2	29	051	2	49	111	2	69	151	2	89	211	2	A9	251	2	C9	311	2	E9	351	2
0A	012	3	2A	052	3	4A	112	3	6A	152	3	8A	212	3	AA	252	3	CA	312	3	EA	352	3
0B	013	3	2B	053	11	4B	113	19	6B	153	27	8B	213	35	AB	253	43	CB	313	51	EB	353	59
0C	014	1	2C	054	1	4C	114	1	6C	154	1	8C	214	1	AC	254	1	CC	314	1	EC	354	1
0D	015	2	2D	055	2	4D	115	2	6D	155	2	8D	215	2	AD	255	2	CD	315	2	ED	355	2
0E	016	3	2E	056	3	4E	116	3	6E	156	3	8E	216	3	AE	256	3	CE	316	3	EE	356	3
0F	017	4	2F	057	12	4F	117	20	6F	157	28	8F	217	36	AF	257	44	CF	317	52	EF	357	60
10	020	1	30	060	1	50	120	1	70	160	1	90	220	1	B0	260	1	D0	320	1	F0	360	1
11	021	2	31	061	2	51	121	2	71	161	2	91	221	2	B1	261	2	D1	321	2	F1	361	2
12	022	3	32	062	3	52	122	3	72	162	3	92	222	3	B2	262	3	D2	322	3	F2	362	3
13	023	5	33	063	13	53	123	21	73	163	29	93	223	37	B3	263	45	D3	323	53	F3	363	61
14	024	1	34	064	1	54	124	1	74	164	1	94	224	1	B4	264	1	D4	324	1	F4	364	1
15	025	2	35	065	2	55	125	2	75	165	2	95	225	2	B5	265	2	D5	325	2	F5	365	2
16	026	3	36	066	3	56	126	3	76	166	3	96	226	3	B6	266	3	D6	326	3	F6	366	3
17	027	6	37	067	14	57	127	22	77	167	30	97	227	38	B7	267	46	D7	327	54	F7	367	62
18	030	1	38	070	1	58	130	1	78	170	1	98	230	1	B8	270	1	D8	330	1	F8	370	1
19	031	2	39	071	2	59	131	2	79	171	2	99	231	2	B9	271	2	D9	331	2	F9	371	2
1A	032	3	3A	072	3	5A	132	3	7A	172	3	9A	232	3	BA	272	3	DA	332	3	FA	372	3
1B	033	7	3B	073	15	5B	133	23	7B	173	31	9B	233	39	BB	273	47	DB	333	55	FB	373	63
1C	034	1	3C	074	1	5C	134	1	7C	174	1	9C	234	1	BC	274	1	DC	334	1	FC	374	1
1D	035	2	3D	075	2	5D	135	2	7D	175	2	9D	235	2	BD	275	2	DD	335	2	FD	375	2
1E	036	3	3E	076	3	5E	136	3	7E	176	3	9E	236	3	BE	276	3	DE	336	3	FE	376	3
1F	037	8	3F	077	16	5F	137	24	7F	177	32	9F	237	40	BF	277	48	DF	337	56	FF	377	64

(Continued from page 16)

Chapter IV
INPUT–OUTPUT TRANSFER OPERATIONS

1. Introduction

Mini-computers find their main applications in the field of process control and data acquisition and measurement. For these applications it is essential that the computer is able to communicate with the peripherals of the control or measurement system.
The exchange of information between the computer and the peripherals is controlled by:
- either a programme stored in the memory of the computer
- or certain peripheral components.

The I/O transfer operations controlled by the computer are called *"programmed I/O operations"*; it is these which are of most interest to us.

2. Programmed I/O transfer operations

2.1. *Necessary conditions*

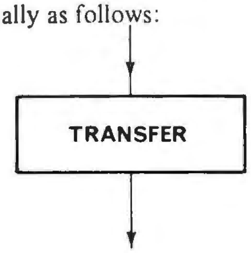
Communication with a peripheral can take place if it is possible to:
- receive and test the information describing the state of the peripheral. Is the magnetic tape ready to record information or is it still being rewound?
- send an item of data from the computer to the peripheral.
- receive an item of data from the peripheral.

These conditions dictate the structure of a peripheral or rather of the CPU-peripheral interface; the I/O system or interface must be provided with:
- a status register allowing control of the peripheral,
- a data register allowing exchange of the data.

A programmed transfer can be carried out in three ways which will now be described.

2.2. *Unconditional transfer*

This method is rarely used because it necessitates exact knowledge of the timing of the process; the data are transferred on the assumption that the peripheral is always ready; this assumption implies that it is superfluous to test the status register of the peripheral.
This transfer may be shown schematically as follows:

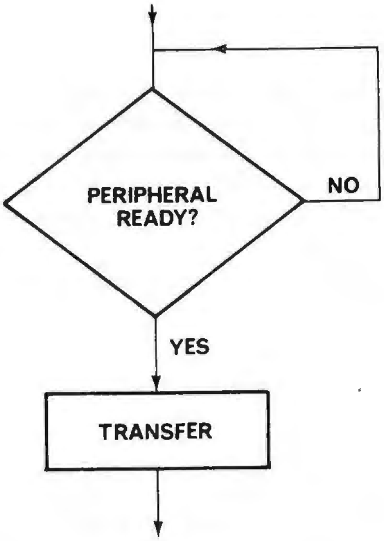


2.3. *Conditional transfer*

This second method is very widely used. The transfer takes place under the control of the programme only when the peripheral is ready to communicate.

The method consists of testing the status register of the peripheral and waiting until it is available. When the status register indicates that this is so, the computer transfers data to or reads it from the Data Register of the peripheral.

Conditional transfer may be schematically represented as follows:



The main disadvantage of this method lies in the fact that it is necessary to wait for the peripheral to become available, resulting in loss of time for the computer.

2.4. *Transfer using an interruption*

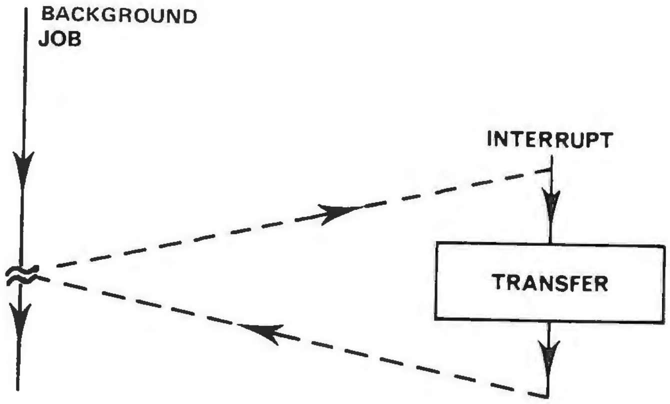
This is the most efficient type of transfer as regards saving computer time. It is also controlled by programme but the computer does not have to continuously test the status register of the peripheral and wait for it to become available.

The computer can be executing a programme, known as the "background job", in the normal way. When the peripheral is ready to effect a transfer it asks to interrupt the computer, making its request on a special line of the processor termed "Interrupt Request Line". The CPU then interrupts its background job and indicates to the other peripherals that it is about to perform an interrupt routine or programme; it indicates this condition by setting to 1 the Interrupt bit of the status *of the CPU* (see part two, Chapter II, 1.).

Then the CPU, which has momentarily dropped its background job, will carry out the transfer routine appropriate to the peripheral; the CPU is said to *"serve"* the peripheral.

Once the transfer has taken place (using the programmed service routine) the CPU resets the Interrupt bit to 0 and carries on with its background job at the point where it left off.

This transfer may be represented schematically as follows:



2.5. *Multiple interrupt structure*

In numerous applications, several peripherals are connected to the same computer. In such a configuration, all the interrupt request lines of the peripherals are OR-wired to the single interrupt request line of the computer.

In this case it is clear that the computer must:

– identify the peripheral which has requested the interruption;

– perform the service routine for this peripheral.

The first requirement allows the assignment of priorities to the interrupt requests of the peripherals; in effect, when a peripheral requests an interruption it does two things:

a) it sets to 1 a special flip-flop associated with it which is called a *"flag flip-flop"*: each peripheral has its own flag;

b) it makes its request on its interrupt request line (physically, it brings this line to the logical state 1).

The computer therefore has to identify the peripheral which has requested an interruption and it does this according to a programme, The CPU must make an "inquiry" referred to as *polling*.

Let us suppose that two peripherals A and B have made their requests simultaneoulsly; the fact that A's flag is tested by the programme *before* that of B allows a higher priority of *interrupt request* to be assigned to A than to B.

It is also possible to assign a priority to the service routines of the peripherals by means of the Interrupt flip-flop (I) of the status register of the CPU.

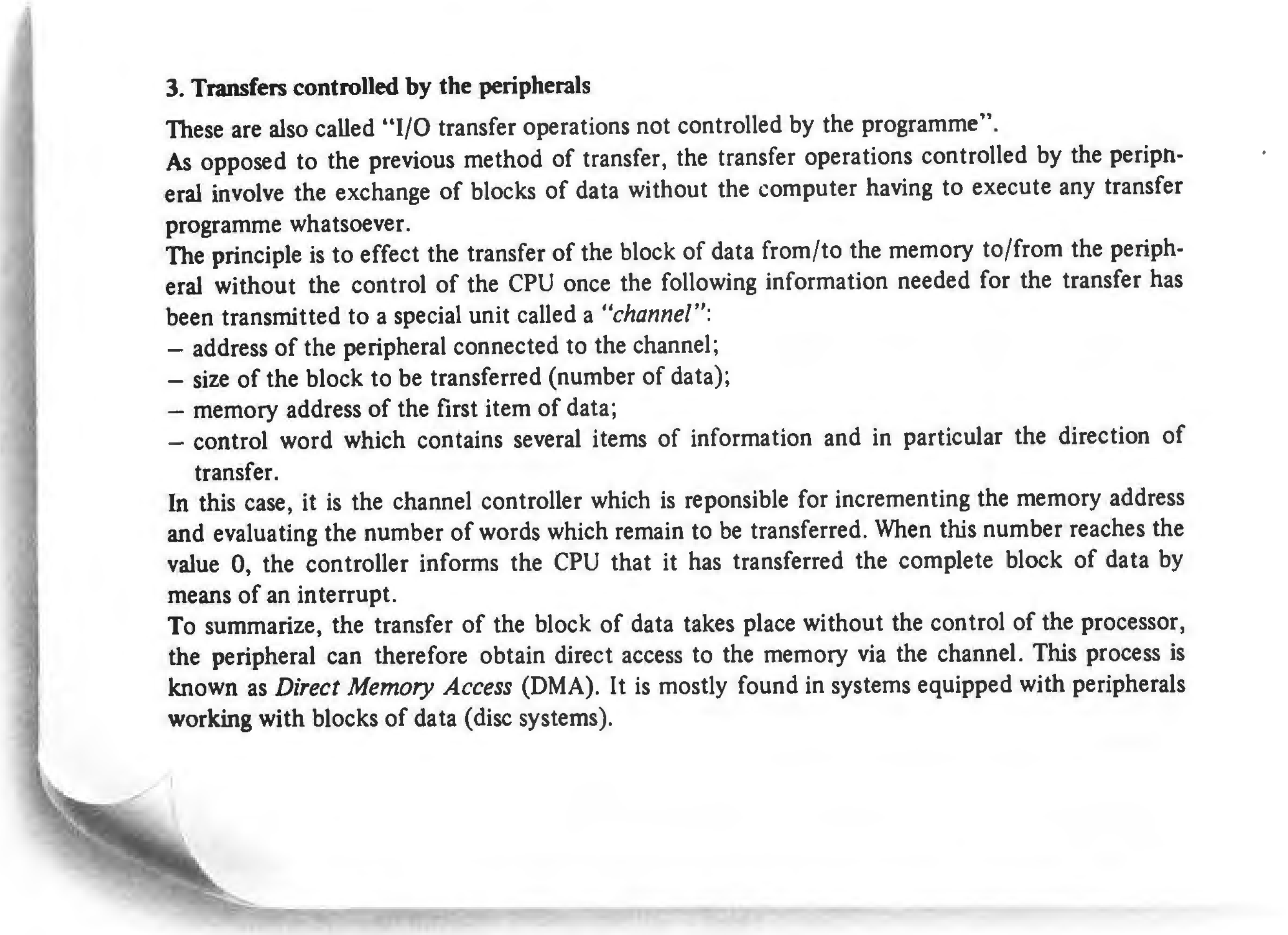
It has been seen above (2.4) that the CPU sets to 1 the bit I of its status every time that it serves an interruption, indicating to the other peripherals that their interrupt requests can not be granted for the moment.

Let us suppose that A has been able to interrupt the main programme of the computer; execution of the service routine of A is then undertaken and the bit I has the value 1 (the requests of the other peripherals are *"masked"*). If the service routine of A contains an instruction resetting to 0 the bit I of the status, it is clear that the service routine of A could in turn be interrupted by an interrupt request of peripheral B; in this cas, B is said to have a *service priority* greater than that of A.

Two things must therefore be clearly distinguished:

– the priority of the request;

– the priority of the service routine.



3. Transfers controlled by the peripherals

These are also called "I/O transfer operations not controlled by the programme".

As opposed to the previous method of transfer, the transfer operations controlled by the peripneral involve the exchange of blocks of data without the computer having to execute any transfer programme whatsoever.

The principle is to effect the transfer of the block of data from/to the memory to/from the peripheral without the control of the CPU once the following information needed for the transfer has been transmitted to a special unit called a *"channel"*:

- address of the peripheral connected to the channel;
- size of the block to be transferred (number of data);
- memory address of the first item of data;
- control word which contains several items of information and in particular the direction of transfer.

In this case, it is the channel controller which is reponsible for incrementing the memory address and evaluating the number of words which remain to be transferred. When this number reaches the value 0, the controller informs the CPU that it has transferred the complete block of data by means of an interrupt.

To summarize, the transfer of the block of data takes place without the control of the processor, the peripheral can therefore obtain direct access to the memory via the channel. This process is known as *Direct Memory Access* (DMA). It is mostly found in systems equipped with peripherals working with blocks of data (disc systems).

(TO BE CONTINUED NEXT MONTH)

MAKINGS OF A MINI

(Continued from page 25)

which generated a short on the mother board.

Though the overall experience of building the computer was rewarding and worthwhile, I would make a few suggestions to the people at MITS: insert the errata sheets closer to the assembly instructions they pertain to in the manual, try to find an alternative—like flat flex cables with integral connectors—to the loose-wire interconnection scheme, and more carefully test the components found in the kit.

Construction of the Altair 8800 produces what some industry observers call a "human" machine. During the assembly process, you get your fingerprints on every component and learn the details of microcomputer architecture by the very order in which you construct the computer.

Unless you buy extra memory, interface cards, peripherals and a high-level software package—all available options—you'll have to learn machine language, enter your programs via the front-panel switch and see the results on the status lights. But machine language programming has its own advantages. In learning it, you can overcome the software-shyness that many microprocessor users seem to exhibit. ◘

Martin Himmelfarb

BITS & BYTES

(Continued from page 40)

tion) and up to 65K bytes of memory. Each node can communicate with eight other nodes through shared memory. The idea is to have each node working on a separate part of a complex problem, such as weather prediction. Each processor won't be very fast, but taken as a whole, it could be incredibly powerful. The number of nodes can vary from 16 to 256. Ho, ho, ho, take that IBM, CDC, Burroughs, Univac, . . . ◘

Join SCCS Now!!

The Southern California Computer Society is a non-profit, international association whose purpose is to promote and encourage a community interest in the computer arts and sciences. Of particular interest is the application of the computer to education and to the social, scientific and environmental problems of society.

Formed in June 1975 by a handful of computer hobbyists and experimenters, the Society has grown rapidly to include 700 consumers, amateurs and professionals. The Society serves as a focal point through which computer buffs can meet, learn, and share common interests and needs.

WHAT CAN SCCS DO FOR YOU?

- Frequent meetings for the exchange of information and computer equipment.
- A magazine (SCCS INTERFACE) communicating topics and news of interest to members, the community, and other organizations.
- Classes on programming, building computers and the general use of computers.
- Informal technical assistance and advice on anything related to computers.
- Consumer clout through group purchases. Co-op purchasing of popular items at reduced prices.
- A center in development with library, computer center, and laboratory.
- Ombudsman service — to help resolve consumer complaints.

WHO SHOULD JOIN SCCS?

- Anyone who is interested in computers as a hobby, or
- Anyone who wants to be introduced to the world of computers (but does not know where to start), or
- Anyone who has computer related talents, equipment, services to share.

JANUARY MEETING
SUNDAY JANUARY 18, 1976
BUILDING "S" CAFETERIA
TRW SYSTEMS
REDONDO BEACH

DOORS OPEN AT 10:00 AM
COFFEE HOUR 12:00-1:00
MEETING 1:00-4:00

BRING YOUR EQUIPMENT TO SHOW OFF, SELL OR SWAP.

MEMBERSHIP FORM

SOUTHERN CALIFORNIA COMPUTER SOCIETY
P.O. BOX 987
SOUTH PASADENA, CA 91030
(213) 682-3108

NAME ______________________________

ADDRESS ______________________________

CITY ______________________________

STATE______________________ZIP ____________

HOME PHONE (________) __________-______________

BUSINESS PHONE (________) __________-______________

DUES: $10.00/YEAR **PLEASE PRINT PLAINLY**

MAKE CHECK PAYABLE TO: SOUTHERN CALIFORNIA COMPUTER SOCIETY

SCCS INTERFACE READER SERVICE

For your convenience, you may receive further product information by circling the appropriate advertiser index numbers and returning the form to:

INTERFACE HOTLINE
P. O. BOX 1234
CERRITOS, CALIF. 90701

NO.	ADVERTISER	PAGE
1	AUTOMATED COMPUTER SYSTEMS	Pg. 59
2	BYTE MAGAZINE	Pg. 30
3	COMPUTERLAND	Pg. 59
4	COMPUTERMART	Pg. 3
5	ELECTRONIC PRODUCT ASSOCIATES, INC.	Pg. 22
6	L.D. BEVEN CO.	Pg. 59
7	LIBE CO.	Pg. 59
8	MITS	Pg. 17
9	SCCS	Pg. 57
10	SPHERE	Pg. IV
11	TECHNOLOGY SERVICE CORP.	Pg. 59
12	THE COMPUTER STORE	Pg. 37
13	THE WIZZIT	PG. 59

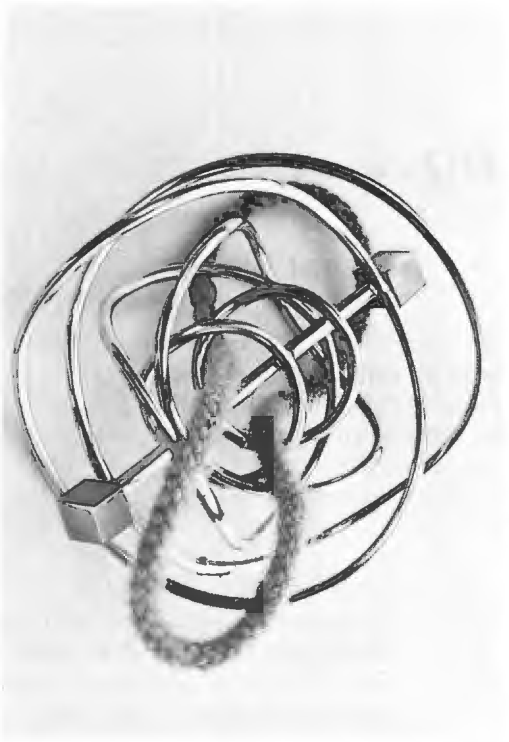
INTERFACE HOTLINE
P. O. Box 1234
Cerritos, CA 90701
Please print or type.

JAN 1976
ADVERTISER NO.
1 2 3 4 5 6 7
8 9 10 11 12 13

Name________________________________

Address______________________________

City________________State__________Zip______



Classified Advertising

Unclassified Advertising.....

Caveat Emptor

Each month SCCS Interface will devote this space to free, non-commercial advertising by members. This is done as a service of the Southern California Computer Society and to help promote communications between computer experimenters and hobbyists throughout the world.

MICROCOMPUTER HELP WANTED FOR PART OF THE ACTION
—Part-time OK—

- *Microcomputer hardware Design Engineer*
- *Microcomputer software Development Engineer*
- *Microcomputer systems Applications Engineer*
- *Microcomputer Sales Engineer*

AUTOMATED COMPUTER SYSTEMS
2361 E. Foothill Blvd., Pasadena, Calif. 91107 (213) 792-1193.

ALTAIR SYSTEM I in a Cybercom Case $400. Or Will trade for HP 65 or unassembled Altair kit. Ken McGinnis, Box 2078, San Mateo, CA 94401.

HELP! I have a Viatron model 2101 that doesn't work. Would appreciate contact with anyone who could provide technical information or programming assistance. Please call Mike Dent (213) 677-5433.

POWER SUPPLY I have a quantity of 5v-6amp highly regulated power supplies taken from keyboard terminals. They also supply 12v, 48v, and 200v each at 1 amp. Full schematics supplied and plans for obtaining -9v, -12v, -5v. $25 plus postage on 15 lbs. and 6% in California. Grant Runyan, 1146 Nirvana Rd., Santa Barbara, CA 93101.

WANTED: Flex-O-Writer manuals & schematics. Please contact Louis G. Fields, 11662 Sunset Blvd., Los Angeles 90049. Phone (213) 472-0388.

EXCELLENT OPPORTUNITY for over the counter sales of Microprocessors, Kits, Peripherals and software. Individual selected must be capable of guiding customers through selection of available equipment and organizing after sale training sessions for hardware and software utilization. Position requires strong background in microprocessor technology and sales experience. Send Resume and salary history to: P. O. Box 2208, Culver City, California 90230.

"HANDY DANDY" PAPER TAPE WINDER $15. Bill Roch, 5133 Catalon Ave., Woodland Hills 91364 (213) 348-4278 or (213) 887-4944.

FOR SALE — FLOPPY DISK DRIVE disassembled w/partial documentation. Will require knowledge of floppies. $100 or trade for 4K static Altair compatible memory. Phone Art Childs (213) 243-5179.

NUMBER OF DAYS PROGRAM How many days til Christmas? Between 1/15/74 and 6/30/76? This little Altair program will provide the answer. I am using the sub-routine in a scheduling system to tell how many days are left to meet a schedule date. If you want a write up on the logic and a paper tape send a buck to Bill Roch, 5133 Catalon Ave., Woodland Hills, Calif. 91364.

WHO NEEDS IBM? Are you using a DEC system-10? If you are interested in starting a club solely devoted to the 10, please call Ralph Klestadt at 784-8319 or 789-9616 after 4 p.m. Remember, you are not alone out there with your 10!

ALTAIR 8800 KIT: CPU & 256 Word memory boards assembled, tested and guaranteed for 3 months [parts, labor & return postage] $459. Manley Nichols, Box 344, Henning, Minn. 56551.

RPC-400 COMPUTER WITH PUNCH, READER and typewriter. Best offer. 1 core $15. 8 x 1K core memory with PS $35. Varian chassis with power supply $50. Several readers, printers, tape reqinds $15 more or less. Need varian DM-122 printed circuit board. Call (213) 796-8270, Bill Pfeiffer.